

Pre-dict
responses



US006092116A

United States Patent [19]

Earnest et al.

[11] Patent Number: 6,092,116

[45] Date of Patent: Jul. 18, 2000

[54] DMA CONTROLLER WITH RESPONSE MESSAGE AND RECEIVE FRAME ACTION TABLES

[75] Inventors: Tim Earnest, Vadnais Heights; Chris Sonnek, North St. Paul, both of Minn.

[73] Assignee: LSI Logic Corporation, Milpitas, Calif.

[21] Appl. No.: 08/761,986

[22] Filed: Dec. 11, 1996

[51] Int. Cl.⁷ G06F 13/00; G06F 13/28

[52] U.S. Cl. 709/236; 709/212; 709/213; 709/214; 709/245; 710/5; 710/22; 710/23; 710/30

[58] Field of Search 395/841-846, 395/821, 823-827, 200.75, 200.76, 200.81, 838, 872-880; 709/212-215, 236-237, 245-246, 250; 710/1, 3-7, 21-26, 30

[56] References Cited

U.S. PATENT DOCUMENTS

4,847,750	7/1989	Daniel	710/25
5,163,136	11/1992	Richmond	710/30
5,414,813	5/1995	Shiobara	709/245
5,542,110	7/1996	Minagawa	710/107
5,619,497	4/1997	Gallagher et al.	370/394
5,717,952	2/1998	Christiansen et al.	710/22
5,809,341	9/1998	Nimishakvi et al.	710/60
5,822,618	10/1998	Ecclesine	710/57

OTHER PUBLICATIONS

"MU95C5480 LANCAM," *Music Semiconductors*, Rev. 0, pp.1-12. (Jul. 22, 1994).

"Information technology -Telecommunications and information exchange between systems -High-level data link control (HDLC) procedures -Elements of procedures," *ISO/IEC 4335*, Fifth edition. (Dec. 15, 1993).

"Information technology -Telecommunications and information exchange between systems -High-level data link control (HDLC) procedures -Frame structure", *ISO/IEC 3309*, Fifth edition. (Dec. 15, 1993).

W. Simpson, Editor, Daydreamer, "PPP in HDLC-like Framing," pp. 1-25, (last modified Jul. 1994) <ftp://ds.INTERNIC.net/rfc>.

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

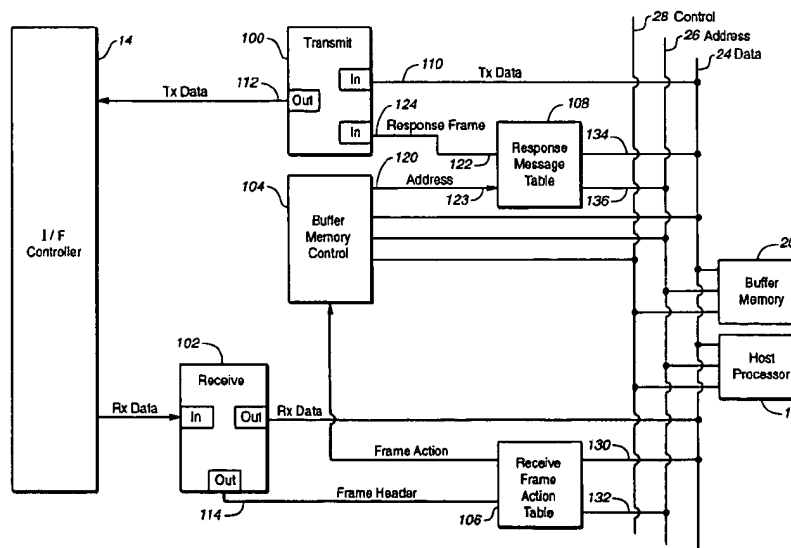
Attorney, Agent, or Firm—Westman, Champlin & Kelly, P.A.

[57] ABSTRACT

A direct memory access (DMA) controller transmits and receives formatted data frames having frame headers in a communications subsystem. The DMA controller includes a transmit input and output, a receive input and output, transmit circuitry, receive circuitry, a receive frame header capture circuit, a receive frame action table and a response message table. The transmit circuitry receives transmit data frames on the transmit input and applies the transmit data frames to the transmit output. The receive circuitry receives receive data frames on the receive input and applies the receive data frames to the receive output. The receive frame header capture circuit obtains a frame header from the received data frames and applies the frame header to the receive frame action table. The receive frame action table generates a frame action command based on the frame header field. The response message table has an address input coupled to receive the frame action command, a response frame output coupled to the transmit circuitry and a plurality of addressable memory locations for storing predetermined response data frames.

Comm. system frame headers
response msg. table

26 Claims, 13 Drawing Sheets



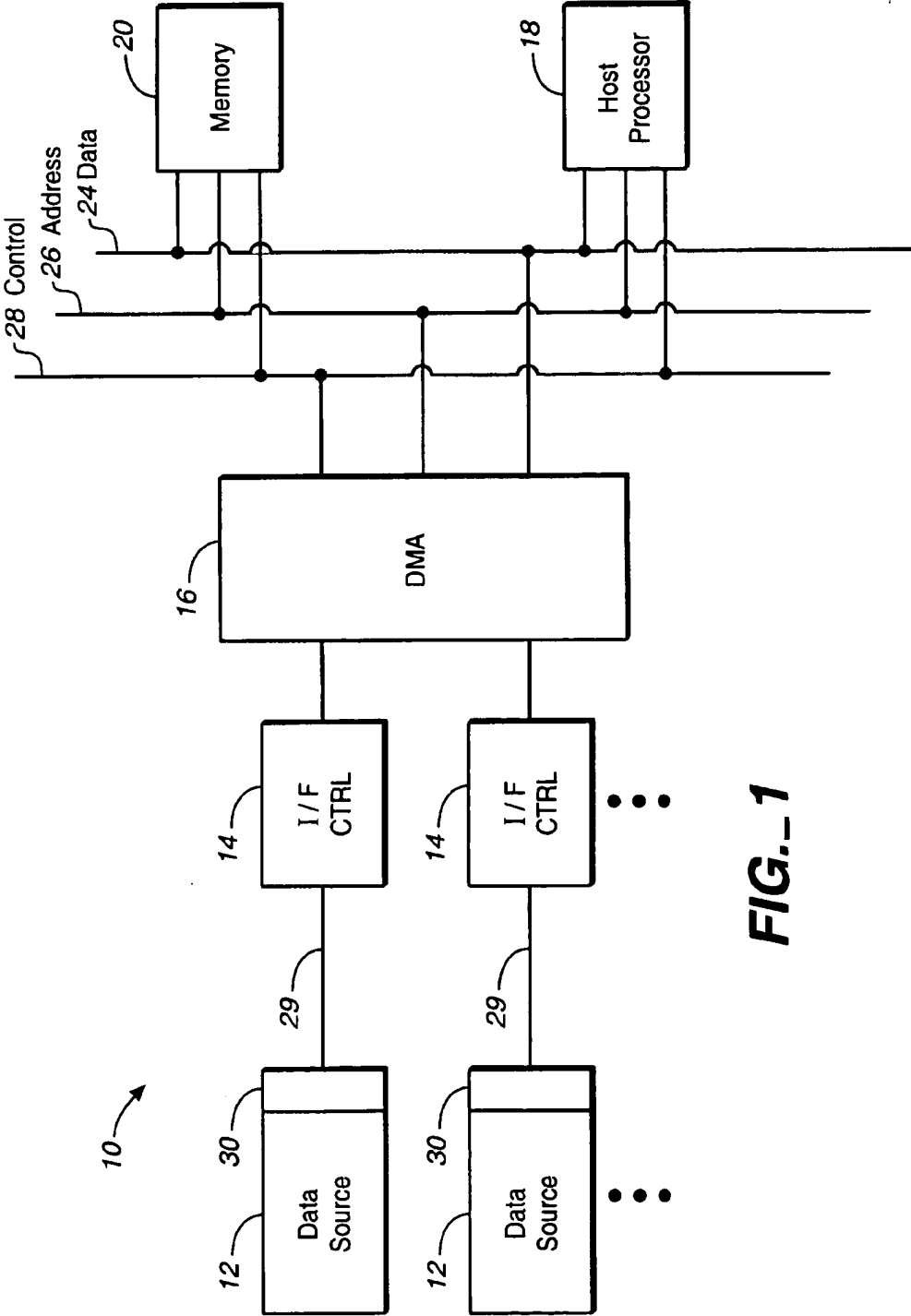


FIG. 1

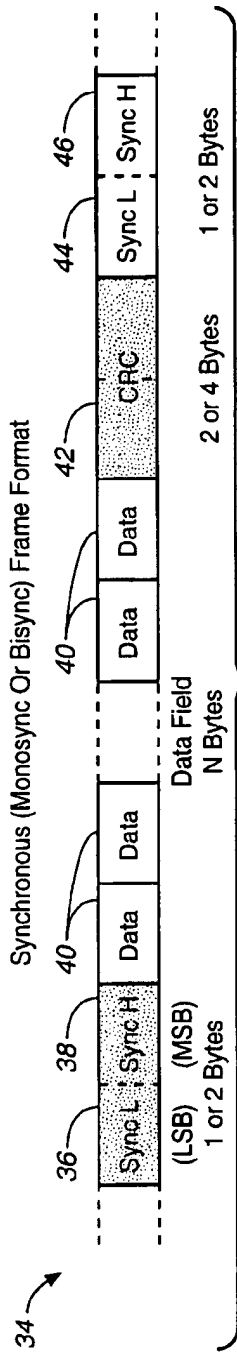


FIG. 2

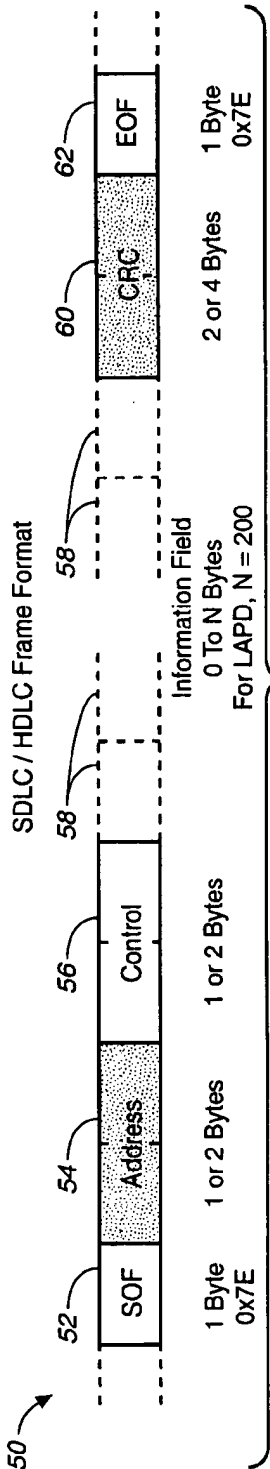


FIG. 3

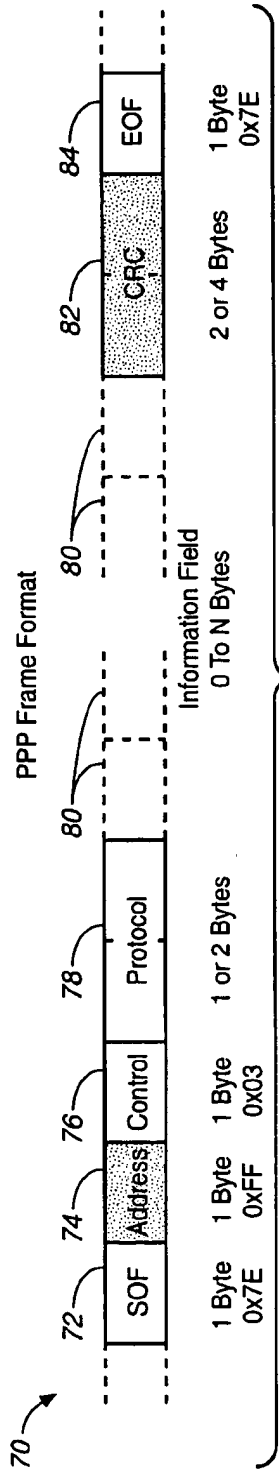


FIG. 4

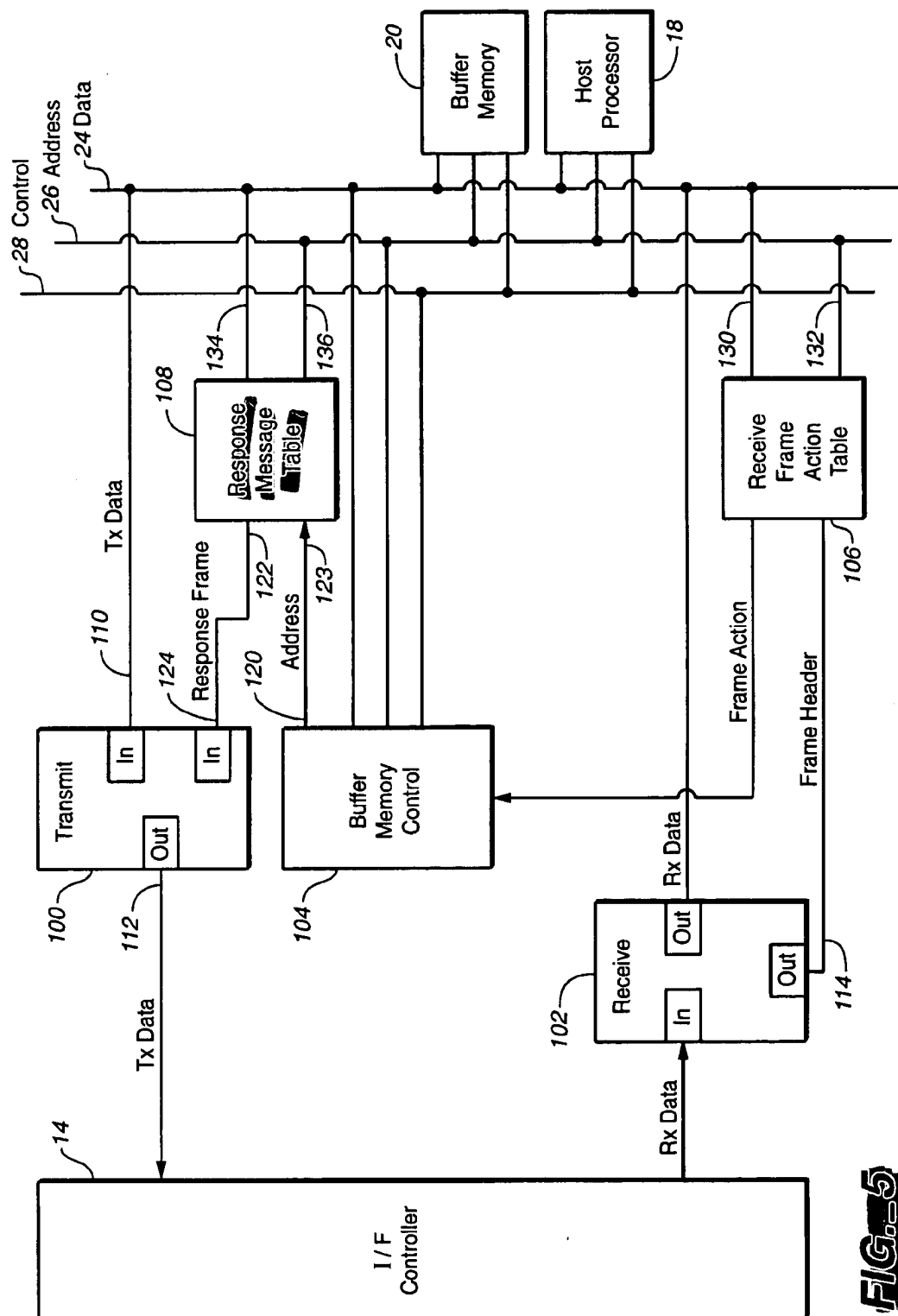
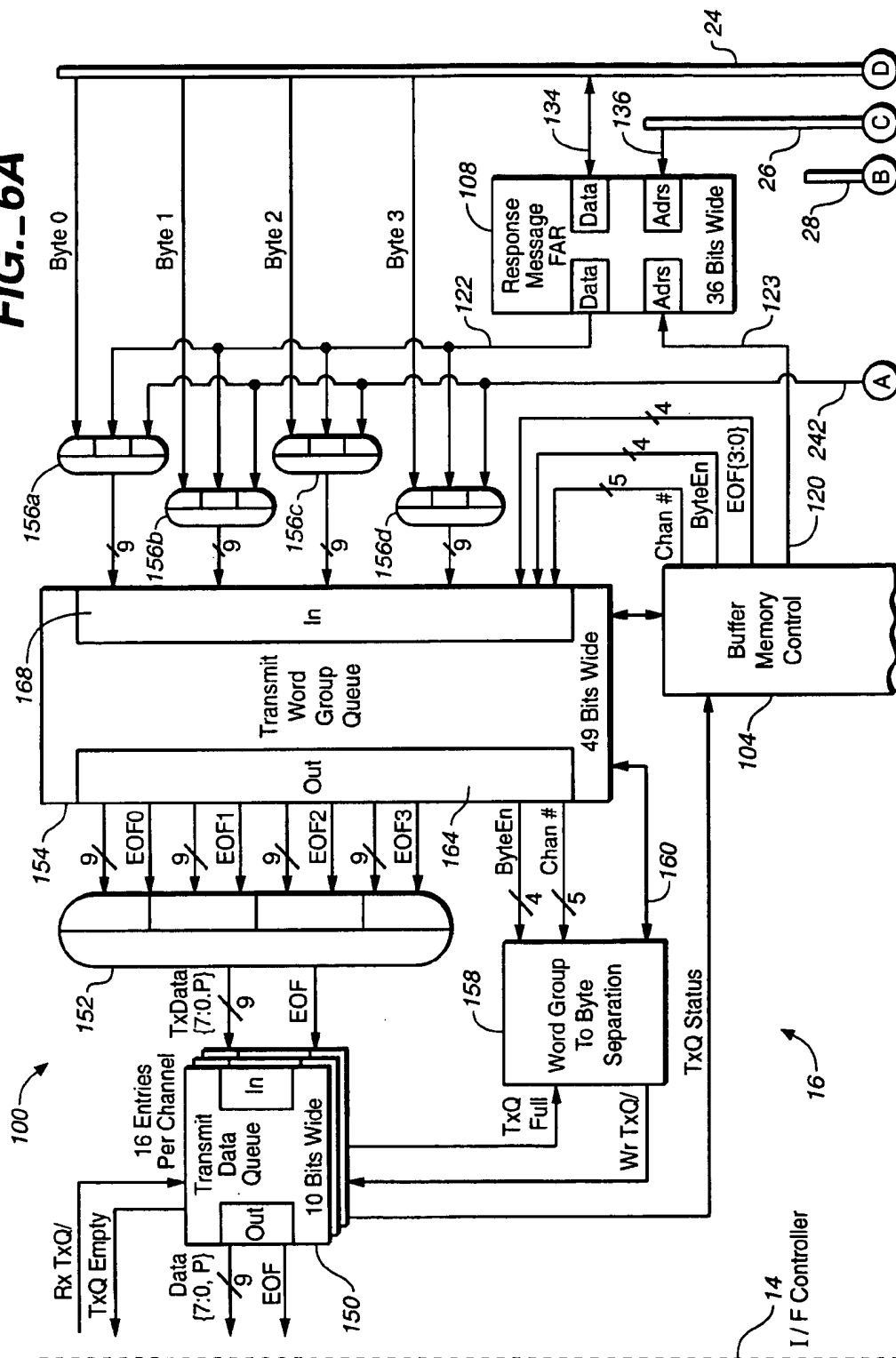
**FIG. 5**

FIG._6A



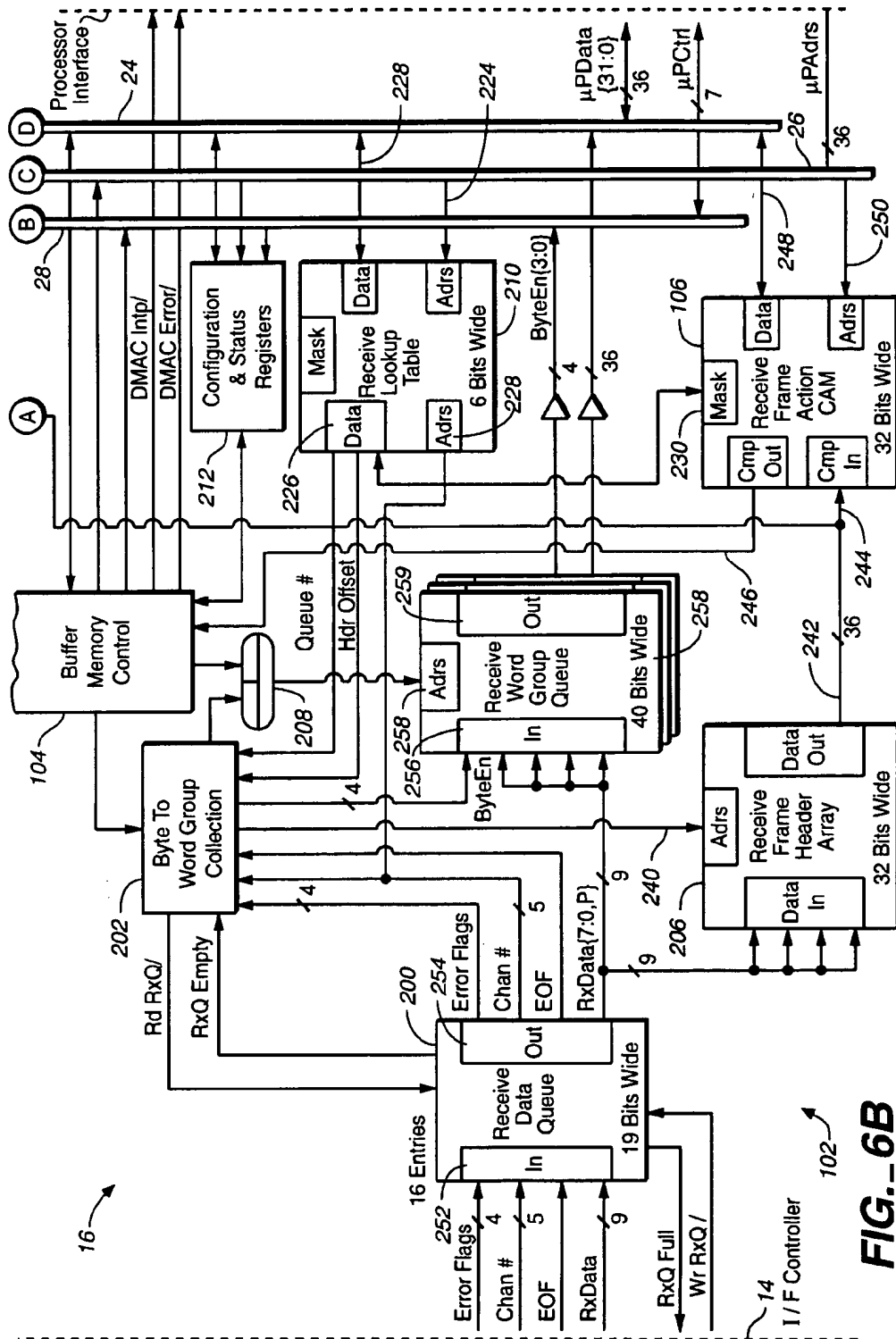


FIG. 6B

Queue # Lookup Table (Adrs 0x1000 - 0x107C) Definitions

Hex Adrs	Description	msb 31	Byte 0	24	23	Byte 1	16	15	Byte 2	8	7	Byte 3	lsb 0
1000	Chan 0 Lookup					CAM Mask			Hdr Offset			Queue #	
1004	Chan 1 Lookup					CAM Mask			Hdr Offset			Queue #	
...	...												
107C	Chan 31 Lookup					CAM Mask			Hdr Offset			Queue #	

FIG._7

Receive Frame Action CAM (Adrs 0x2000 - 0x21FC) Definitions

Hex Adrs	Description	msb 31	Byte 0	24	23	Byte 1	16	15	Byte 2	8	7	Byte 3	lsb 0
2000	Locn 0 Cmp In		Byte 0			Byte 1			Byte 2			Byte 3	
2004	Locn 0 Cmp Out											Frame Action	
...	...												
21F8	Locn 63 Cmp In		Byte 0			Byte 1			Byte 2			Byte 3	
21FC	Locn 63 Cmp Out											Frame Action	

FIG._8

108

Response Message FAR (Adrs 0x3000 - 0x307C) Definitions

Hex Adrs	Description	Wr / Rd	msb 31	Byte 0 24	Byte 1 16	Byte 2 8	Byte 3 0	lsb
3000	Response Data	Wr / Rd	Byte 0	Byte 1	Byte 2	Byte 3	0	
...	...							
307C	Response Data	Wr / Rd	Byte n-3	Byte n-2	Byte n-1	Byte n		

Response Message FAR

FIG._9

260

DMAC Configuration Register (Adrs 0x000) Bit Definitions

Hex Adrs	Description	Wr / Rd	msb 31	Byte 0 24	Byte 1 16	Byte 2 8	Byte 3 0	lsb
0000	DMAC Config Reg	Wr / Rd					0 0 0 r 0 b t r	

Reset Hardware
1 = Reset Asserted, 0 = Reset Inactive

Burst Mode On Processor Bus
1 = Enabled, 0 = Disabled

Transmit Section
1 = Enabled, 0 = Disabled

Receive Section
1 = Enabled, 0 = Disabled

FIG._10

Channel Enable Register (Adrs 0x0004) Bit Definitions

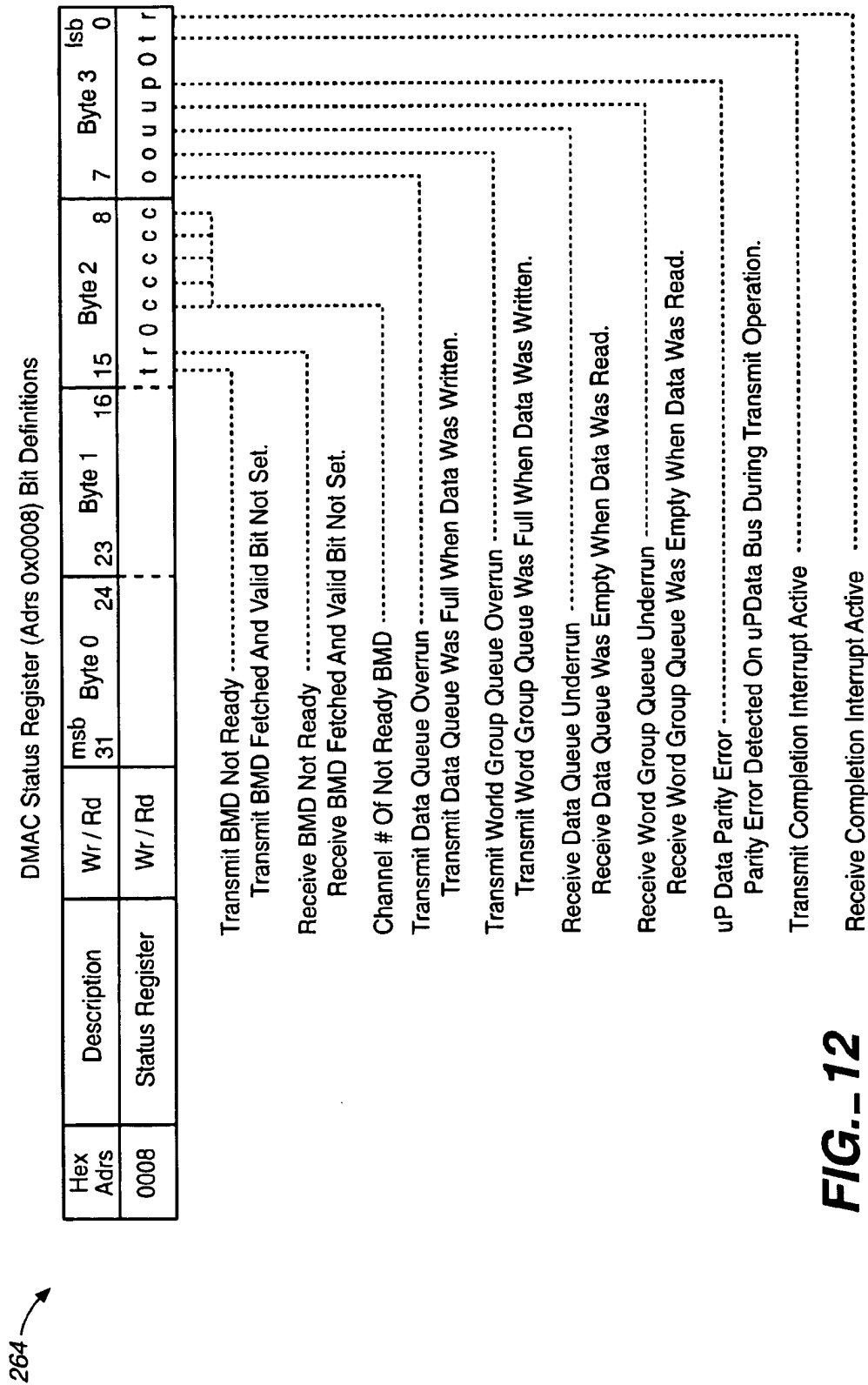
Hex Adrs	Description	Wr / Rd	msb 31	Byte 0 24	Byte 1 23	Byte 2 16	Byte 3 8	Byte 7 7	Isb 0
0004	Chan Enable Reg	Wr / Rd	e e e e e e e e	e e e e e e e e	e e e e e e e e	e e e e e e e e	e e e e e e e e	e e e e e e e e	e e e e e e e e

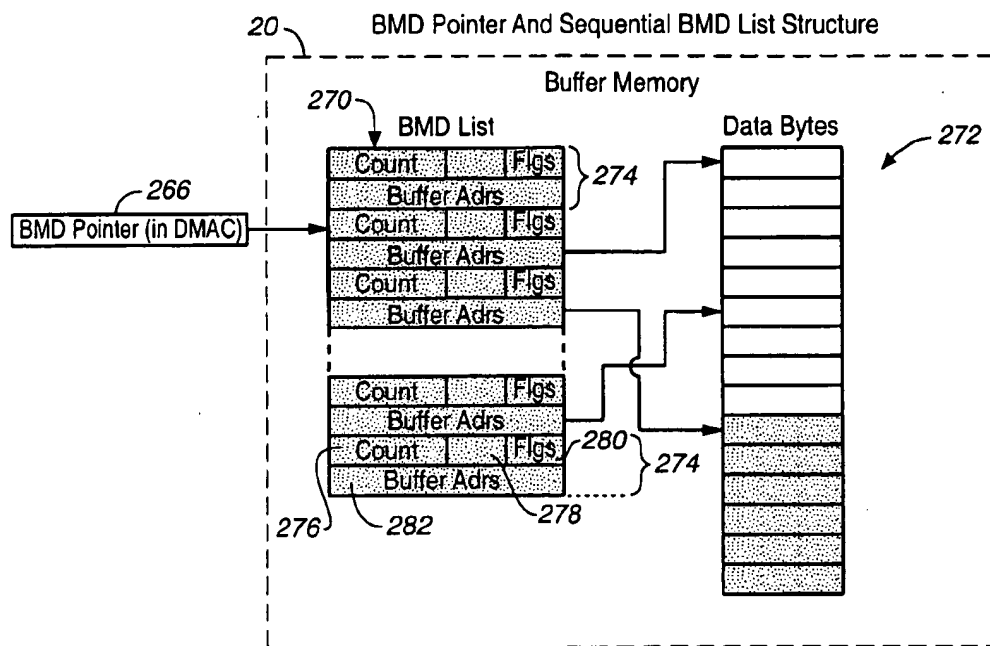
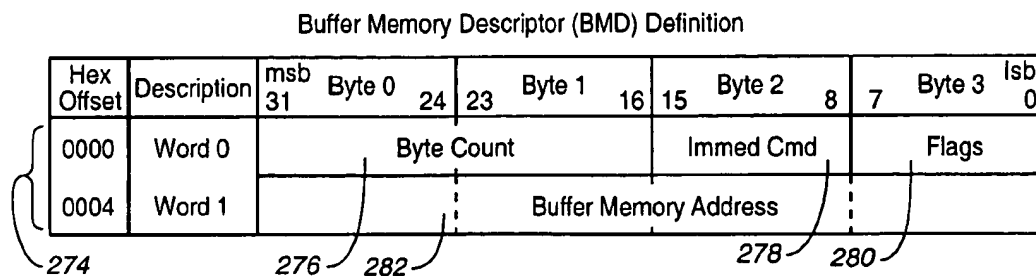
FIG. 11

BMD Pointer Registers (Adrs 0x0100 - 0x01FC) Definitions

Hex Adrs	Description	Wr / Rd	msb 31	Byte 0 24	Byte 1 23	Byte 2 16	Byte 3 8	Byte 7 7	Isb 0
0100	Chan 0 Tx Pntr	Wr / Rd	Transmit Channel 0 Buffer Memory Descriptor Pointer						
0104	Chan 1 Tx Pntr	Wr / Rd	Transmit Channel 1 Buffer Memory Descriptor Pointer						
⋮	⋮								
017C	Chan 31 Tx Pntr	Wr / Rd	Transmit Channel 31 Buffer Memory Descriptor Pointer						
0180	Chan 0 Rx Pntr	Wr / Rd	Receive Channel 0 Buffer Memory Descriptor Pointer						
0184	Chan 1 Rx Pntr	Wr / Rd	Receive Channel 1 Buffer Memory Descriptor Pointer						
⋮	⋮								
018C	Chan 31 Rx Pntr	Wr / Rd	Receive Channel 31 Buffer Memory Descriptor Pointer						

FIG. 13



**FIG. 14****FIG. 15**

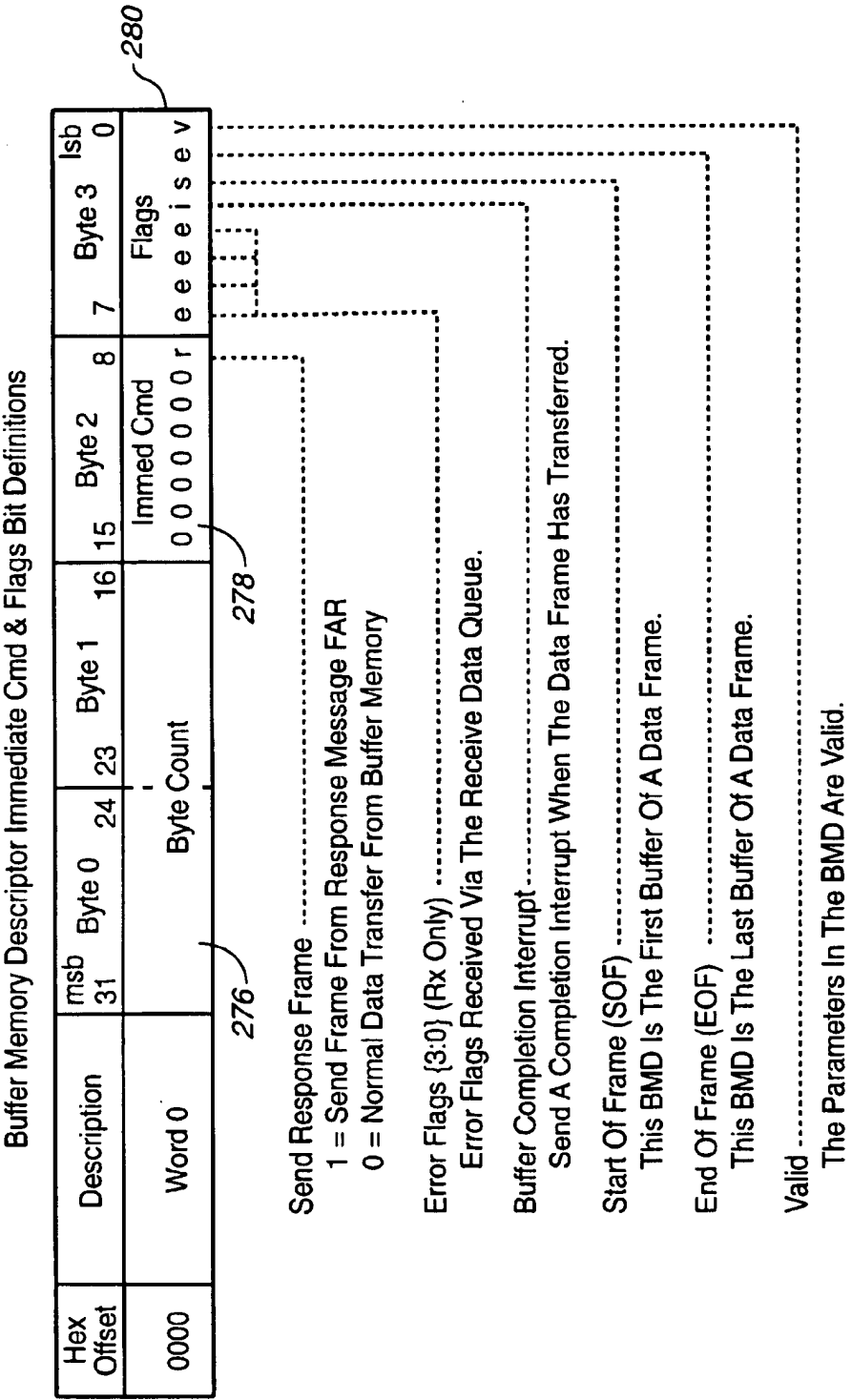
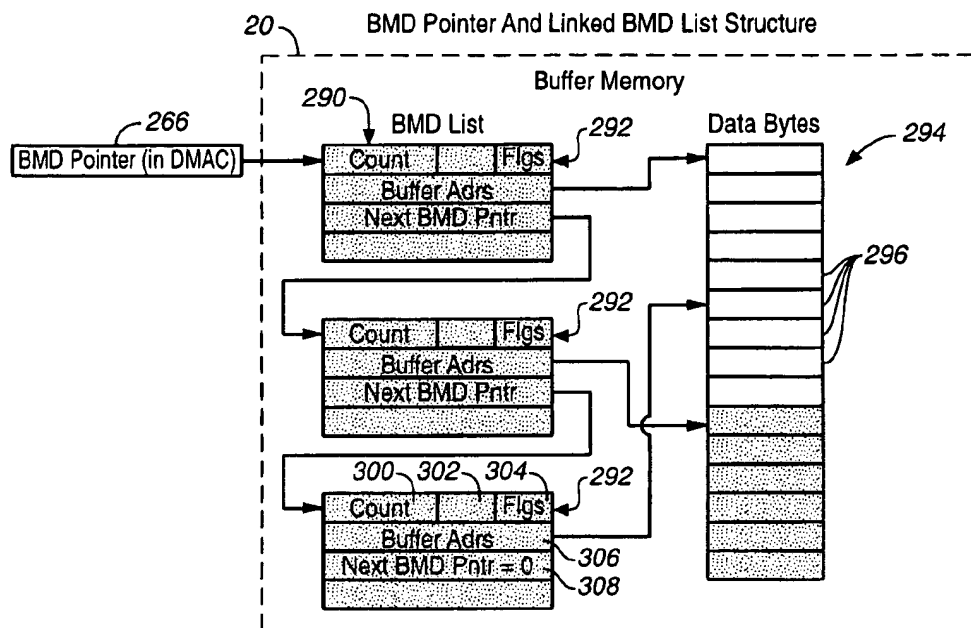
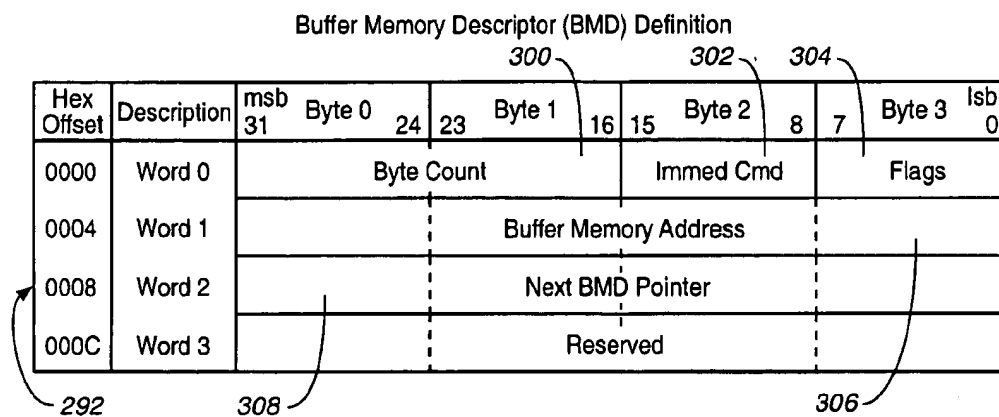


FIG._16

**FIG. 17****FIG. 18**

Buffer Memory Descriptor Immediate Cmd & Flag Bit Definitions

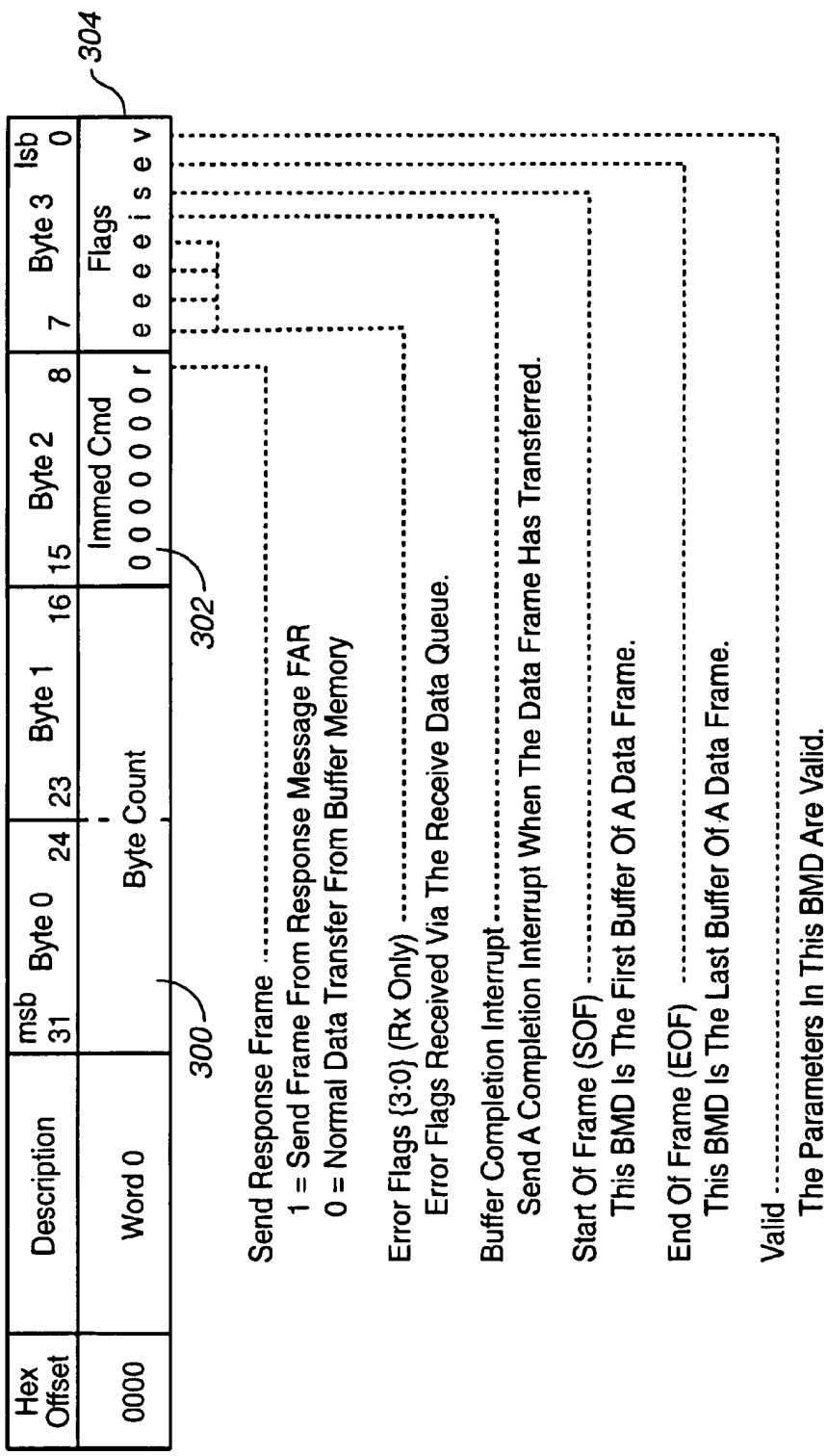


FIG. 19

DMA CONTROLLER WITH RESPONSE MESSAGE AND RECEIVE FRAME ACTION TABLES

BACKGROUND OF THE INVENTION

The present invention relates to digital communications circuits and, more particularly, to a Direct Memory Access (DMA) controller having enhanced data frame processing circuitry.

A typical communications subsystem includes a physical layer interface controller, such as a serial Wide Area Network (WAN) controller, which is coupled to a communications line for transmitting data to and receiving data from a remote data source. A DMA controller is coupled between the interface controller and a host processor data bus for communication with a host processor and its associated memory. Data is typically transmitted and received in data frames. Each data frame includes a frame header field and a data field. The frame header field typically includes a frame address field, a frame control field and a frame protocol field.

When the interface controller receives a data frame from the remote data source, the data frame is passed through the interface controller to the DMA controller. The DMA controller writes the data frame to the host processor memory and then notifies the host processor that data has arrived. The host processor then looks at the data frame in the host processor memory and makes a decision on what to do with the data frame and where to route the data frame. The host processor then transfers the data frame to its destination.

High Level Data Link Control (HDLC), Synchronous Data Link Control (SDLC), and Point to Point (PPP) link layer protocols normally send "response" frames in response to "request" data frames. If a request data frame has been received in the host processor memory, the host processor creates a response frame in the host processor memory to acknowledge receipt of the data frame. The host processor notifies the DMA controller to transfer the response frame from the host processor memory to the data source through the interface controller. The DMA controller then transfers the response data frame.

This procedure requires large processing overhead. First, the host processor consumes valuable processing time looking at data frames in the host processor memory and making decisions on what to do with the data frames and where to route the data frames. The host processor also consumes valuable processing time creating response frames and notifying the DMA controller to transfer the response frames. Finally, the DMA controller uses valuable bandwidth on the host processor data bus getting the response frames from the host processor memory. Thus, there is a need for a communications subsystem having increased efficiency and data throughput.

SUMMARY OF THE INVENTION

The direct memory access (DMA) controller of the present invention transmits and receives formatted data frames having frame headers in a communications subsystem. The DMA controller includes a transmit input and output, a receive input and output, transmit circuitry, receive circuitry and a response message table. The transmit circuitry receives transmit data frames on the transmit input and applies the transmit data frames to the transmit output. The receive circuitry receives receive data frames on the receive input and applies the receive data frames to the receive output. The response message table has an address input coupled to the receive circuitry, a response frame

output coupled to the transmit circuitry and a plurality of addressable memory locations for storing predetermined response data frames.

In one embodiment, the DMA controller further includes a frame header capture circuit which captures the frame header of each receive data frame and a receive frame action table which determines an action to be performed on the receive data frame as a function of the captured frame header. When the receive frame action table determines that a response frame should be transmitted from the response message table, the receive frame action table provides the response message table with the address at which the desired response frame is stored. A host processor coupled to the DMA controller can also transmit any response frame from the response message table.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a communications circuit according to one embodiment of the present invention.

FIG. 2 is a diagram illustrating a synchronous data frame format.

FIG. 3 is a diagram illustrating an SDLC/HDLC protocol frame format.

FIG. 4 is a diagram illustrating a PPP protocol frame format.

FIG. 5 is a simplified block diagram of a DMA controller shown in FIG. 1.

FIGS. 6A and 6B together form a more detailed block diagram of the DMA controller shown in FIG. 1 according to one embodiment of the present invention.

FIG. 7 is a diagram which illustrates the bit definitions of a queue # look up table.

FIG. 8 is a diagram which illustrates the bit definitions of a receive frame action table.

FIG. 9 is a diagram which illustrates the bit definitions of a response message table.

FIG. 10 is a diagram illustrating the bit definitions of a configuration register within the DMA controller.

FIG. 11 is a diagram illustrating the bit definitions of a channel enable register within the DMA controller.

FIG. 12 is a diagram illustrating the bit definitions of a DMAC status register.

FIG. 13 is a diagram illustrating the bit definitions of a BMD pointer register.

FIG. 14 is a diagram which illustrates the data structure of a buffer memory having a sequential buffer memory descriptor (BMD) list.

FIG. 15 is a diagram illustrating the bit definitions of a buffer memory descriptor within the sequential BMD list shown in FIG. 14.

FIG. 16 is a diagram illustrating the bit definitions of an immediate command field and flags field within the sequential BMD list shown in FIG. 14.

FIG. 17 is a diagram which illustrates the data structure of a buffer memory having a linked BMD list.

FIG. 18 is a diagram illustrating the bit definitions of a buffer memory descriptor in the linked BMD list shown in FIG. 17.

FIG. 19 is a diagram illustrating the bit definitions of a buffer memory descriptor intermediate command field and flags field in the linked BMD list shown in FIG. 17.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The DMA controller of the present invention is used with communications protocols in which formatted data frames

3

are transmitted from one point to another. Formatted data frames include a frame header field and a data field. A typical frame header field includes a frame address field, a frame control field and a frame protocol field. These fields determine the destination of the data frame and the type of data frame. The DMA controller of the present invention increases efficiency and data throughput by using specialized memory arrays which assist in processing data frames and triggering automatic response frames.

FIG. 1 is a block diagram of a communications circuit according to one embodiment of the present invention. Communications circuit 10 includes a plurality of data sources 12, a plurality of interface controllers 14, a DMA controller 16, a host processor 18, and a host processor buffer memory 20. DMA controller 16, host processor 18, and buffer memory 20 are coupled to one another through host processor data bus 24, address bus 26 and control bus 28. A communications cable 29 is coupled between each data source 12 and its associated interface controller 14. Each data source 12 includes a physical layer interface 30 which has transmit and receive circuitry for communicating over cable 29. In one embodiment, each interface controller 14 includes a serial Wide Area Network (WAN) controller which enables high speed serial communications over several types of synchronous serial channels. However, other types of controllers or physical interfaces can be used with the present invention.

Data frames are passed from data source 12 through cable 29 and interface controller 14 to DMA controller 16. DMA controller 16 writes the received data frame to memory 20 over host processor data bus 24 and then notifies host processor 18 that data has arrived. Host processor 18 then transfers the data frame to its destination, which may be another location within buffer memory 20 or another device coupled to data bus 24. In the case where the received data frame includes a control message, host processor 18 performs an operation that is defined in the control message.

DMA controller 16 relieves host processor 18 from some of its decision making responsibilities by determining what action to take on an incoming data frame based on the frame header field. The action can include discarding the data frame, generating a host processor interrupt or sending a response frame, for example. A response frame can be triggered automatically by DMA controller 16 or DMA controller 16 can be instructed by host processor 18 to send a response frame. In addition, host processor 18 can initiate transmission of data frames from buffer memory 20 through DMA controller 16 and interface controllers 14.

Several communications protocols can be used with the present invention, such as Synchronous Data Link Control (SDLC), High Level Data Link Control (HDLC) and Point to Point (PPP) link layer protocols. In synchronous communications, serial data streams include individual data bits which are separated by bit boundaries. FIG. 2 is a diagram illustrating a generalized data frame format for synchronous data frames. Data frame 34 includes synchronizing fields 36 and 38, data fields 40, Cyclic Redundancy Check (CRC) field 42 and synchronizing fields 44 and 46. For synchronous data, start and stop bits are not used. Instead, the data is transferred in frames with no gaps between characters. Each data frame is separated by either one synchronizing character or two synchronizing characters within fields 36, 38, 44 and 46. The same synchronizing characters are used at the beginning of data frame 34 and at the end of data frame 34, and are used as Start of Frame (SOF) and End of Frame (EOF) characters (e.g. field 36=0x44 and field 38=0x44, where "0x" designates a hexa-

4

decimal number). An EOF character for one data frame can be used as an SOF character for the next data frame.

FIG. 3 is a diagram of a synchronous data frame format according to the SDLC/HDLC protocol. Data frame 50 includes SOF field 52, frame address field 54, frame control field 56, information fields 58, CRC field 60 and EOF field 62. SDLC/HDLC is a bit oriented synchronous protocol which transmits data in frames using one synchronizing character (e.g. field 52=0x7E and field 62=0x7E). The frame begins when the SOF character is received and ends when an EOF character is received. As mentioned above, the EOF character for one frame can be the SOF character for the next frame.

FIG. 4 is a diagram of a synchronous data frame format according to the PPP protocol. Data frame 70 includes SOF field 72, address field 74, control field 76, protocol field 78, information fields 80, CRC field 82 and EOF field 84. PPP is a byte orientated synchronous protocol. PPP transmits data in frames, similar to the HDLC protocol, using one synchronizing character (e.g. field 72=0x7E and field 84=0x7E). The frame begins when an SOF character is received and ends when an EOF character is received. An EOF character for one frame can be the SOF character for the next frame.

FIG. 5 is a simplified block diagram of DMA controller 16. DMA controller 16 includes transmit circuitry 100, receive circuitry 102, buffer memory control circuit 104, receive frame action table 106 and response message table 108. Buffer memory control circuit 104 controls the operation of transmit circuitry 100, receive circuitry 102, receive frame action table 106 and response message table 108 according to internal logic and commands passed from host processor 18 through data bus 24, address bus 26 and control bus 28. In transmit mode, transmit circuitry 100 receives transmit data (labelled "TxData") at input 110 from buffer memory 20 over host processor data bus 24. Transmit circuitry 100 then transmits the data to interface controller 14 at output 112 under the control of buffer memory control circuit 104.

In receive mode, receive data frames (labelled "RxData") are passed from interface controller 14 to receive circuitry 102. Receive circuitry 102 then writes the received data frames to buffer memory 20 over data bus 24, under the control of buffer memory control circuit 104, and notifies host processor 18 that data has arrived. Receive circuitry 102 also identifies the frame header (fields 54 and 56 in FIG. 3 or 74, 76 and 78 in FIG. 4) within the data frame and supplies the frame header to receive frame action table 106 over output 114. Receive frame action table 106 stores a plurality of predetermined frame action fields, which define the actions to be taken when corresponding frame headers have been received. When a frame header is presented at the input of frame action table 106, table 106 provides the corresponding frame action field to buffer memory control circuit 104 to inform the control circuit of the action to take on the newly received data frame. In one embodiment, the frame action field instructs buffer memory control circuit 104 to discard the received data, generate a host processor interrupt or send a response frame from response message table 108.

Response message table 108 stores a plurality of predetermined response data frames. When the frame action field indicates that a response message should be sent, the frame action field includes an address which identifies the desired response message residing in response message table 108. Buffer memory control circuit 104 receives the address and

response
mssg
table

provides the address to response message table 108 at output 120. Response message table 108 receives the address at input 123 and provides the corresponding response data frame on output 122, which is coupled to input 124 of transmit circuitry 100. Transmit circuitry 100 then transmits the response data frame on output 112. If required, host processor 18 has the option of sending its own response data frame back to the data source from host processor buffer memory 20 over data bus 24.

Receive frame action table 106 and response message table 108 are both programmable by host processor 18. Receive frame action table 106 includes data input 130 and address input 132 which are coupled to host processor data bus 24 and address bus 26, respectively. Similarly, response message table 108 includes data input 134 and address input 136 which are coupled to host processor data bus 24 and address bus 26. These inputs allow host processor 18 to initialize or reprogram tables 106 and 108, as desired, which allows host processor 18 to maintain control over all aspects of data frame decision making and response frame generation. The interface with host processor 18 include a MIPS or generic processor interface, for example.

FIGS. 6A and 6B together form a more detailed block diagram of DMA controller 16 according to one embodiment of the present invention in which the DMA controller is adapted to be used with any interface controller core that transmits or receives data into an 8-bit (plus an optional parity bit) queuing interface.

Transmit circuitry 100 is shown in FIG. 6A. Transmit circuitry 100 includes transmit data queue 150, multiplexer 152, transmit word group queue 154, multiplexers 156a-156d and word group to byte separation circuit 158. Each multiplexer 156a-156d simultaneously receives a corresponding byte of data, Byte 0, Byte 1, Byte 2 and Byte 3, from host processor data bus 24 and provides the bytes of data to transmit word group queue 154 when directed by buffer memory control circuit 104. Multiplexers 156a-156d also receive inputs from output 122 of response message table 108 and from output 242 of receive frame header array 206 (shown in FIG. 6B). The outputs of multiplexers 156a-156d are coupled to input 168 of transmit word group queue 154.

Input 168 also receives a 5-bit channel number Chan #, a 4-bit byte enable signal Byte En and a 4-bit End of Frame signal EOF {3:0} from buffer memory control circuit 104, which are matched with the incoming data bytes. Transmit word group queue 154 has an output 164 which provides four bytes of data and their corresponding end of frame bits EOF0-EOF3 to multiplexer 152. The corresponding byte enable signal Byte En and channel number Chan # are routed to word group to byte separation circuit 158. Multiplexer 152 is operated under the control of word group to byte separation circuit 158 which selects one data byte at a time and its corresponding end of frame bit for transmission to transmit data queue 150. The selected data byte and end of frame bit form a transmit data signal TxData{7:0,P} and an end of frame signal EOF.

Transmit data queue 150 includes a plurality of channels with each channel storing up to 16 data bytes, which are ten bits wide, for transmission to the corresponding interface controller 14. The size of each channel and the number of channels can be varied as desired to suit a particular application. Each transmit data byte and its corresponding parity bit and EOF bit are stored as an entry in the appropriate channel of transmit data queue 150.

Word group to byte separation circuit 158 receives the 4-bit byte enable signal Byte En and the 5-bit channel

number Chan # from transmit word group queue 154 for every four bytes of data transferred through multiplexer 152. Word group to byte separation circuit 158 provides a write control signal Wr TxQ/ to each channel of transmit data queue 150. Write control signal Wr TxQ/ is active when the Byte En signal for the corresponding Chan # is active. When write control signal Wr TxQ/ is active, the selected transmit data byte TxData{7:0,P} and EOF bit at the output of multiplexer 152 is stored in the corresponding, enabled channel of transmit data queue 150. Transmit data queue 150 provides a transmit queue full signal TxQ Full to word group to byte separation circuit 158 when the corresponding channel in queue 150 is full. Transmit data queue 150 also provides a transmit queue status signal TxQ Status to buffer memory control circuit 104.

Each channel of queue 150 has a 9-bit data output Data{7:0,P}, an end of frame output EOF and a transmit queue empty status output TxQ Empty which are coupled to the corresponding interface controller 14. When a channel of queue 150 is empty, the TxQ Empty status output for that channel is active. Each channel of queue 150 receives a read transmit queue control signal Rd TxQ/ from the corresponding interface controller 14. The Rd TxQ signal is active when the corresponding interface controller 14 is requesting a data byte to be transmitted from transmit data queue 150. Transmission of data out of queue 150 is asynchronous to writing data into the queue.

Buffer memory control circuit 104 operates transmission circuitry 100 to multiplex data transmission through the various channels. In one embodiment, each channel is given an equal time to share access to buffer memory 20 in order to prevent overruns and underruns in any particular queue channel. This process is achieved in DMA controller 16 using word groups. A word group is a collection of about 1 to 16 bytes of information that are transmitted to or from buffer memory 20. In a preferred embodiment, a full word group includes four words, with each full word including four bytes of data. The actual number of bytes transferred in each word or word group depends on the address and byte count of the data buffer that is being accessed within buffer memory 20. When the buffer address begins on an odd boundary, such as in the middle of a word boundary within a data frame, DMA controller 16 first transfers one to four bytes of information to reach a full word boundary. DMA controller then transfers up to four words to reach a four word (word group) boundary. Once a word group boundary is reached, DMA controller 16 transfers the data in four word bursts, until the last bytes of data within the data frame are transferred. This allows data to be transferred along word boundaries.

Buffer memory control circuit 104 transfers one word group per DMA channel from buffer memory 20 to the transmit word group queue 154 through host processor data bus 24 and multiplexers 156a-156d. The specified Chan # and the four Byte En bits accompany each data word through transmit word group queue 154. While buffer memory control circuit 104 is getting another word group from buffer memory 20, word group to byte separation circuit 158 moves the present word group into the appropriate channel of transmit data queue 150, one byte at a time. A byte is discarded if it does not have its respective Byte En signal active.

Receive circuitry 102 is shown in FIG. 6B. Receive circuitry 102 includes receive data queue 200, byte to word group collection circuit 202, receive word group queue 204, receive frame header array 206, receive frame action table 106, multiplexer 208 and queue # look up table 210. DMA

controller 16 also includes a configuration and status register block 212 which is coupled to buffer memory control circuit 104.

In one embodiment, receive data queue 200 stores up to 16 entries locations, with each entry being 19 bits wide. Only one receive data queue 200 is required for all multiplexed channels. Receive data queue 200 has an input 252 which receives 19 bits of data from an interface controller 14. The 19 bits of data include an 8-bit, plus parity receive data signal RxData{7:0,P}, an end of frame bit EOF, a 5-bit channel number Chan #, and a 4-bit error flag signal Error Flags. Receive data queue also receives a write receive queue control signal Wr RxQ/ from interface controller 14 which is active when receive data is to be written in queue 200.

In return, receive data queue 200 provides a receive queue full status signal RxQ Full to interface controller 14 to indicate when receive data queue 200 is full. Receive data queue 200 has an output 254 at which one byte of stored data RxData{7:0,P} is provided when a read receive queue control signal Rd RxQ/ is received from byte to word group collection circuit 202. The 4-bit Error Flags signal, 5-bit Chan # signal and the EOF bit which correspond to the RxData{7:0,P} signal are also provided on output 254. Receive data queue 200 also generates a receive queue empty status signal RxQ Empty when receive data queue 200 is empty. The RxQ Empty status signal is monitored by byte to word group collection circuit 202.

Receive word group queue 204 is formed of a plurality of individual queues. Each queue has a data input 256, an address input 258 and a data output 259 and is 40 bits wide for receiving an entire 36 bit data frame, including associated parity and Byte En bits. Data input 256 receives the RxData{7:0,P} signal from output 254 of queue 200 and a 4-bit byte enable signal Byte En {3:0} from byte to word group collection circuit 202. Address input 258 receives a queue address from multiplexer 208, which specifies the address in queue 204 that the data will be written. The address can be provided by either byte to word group collection circuit 202 or buffer memory control circuit 104. The address at input 258 is a function of the Chan # signal accompanying the data at output 254 and a Queue # signal provided by queue # lookup table 210. The Queue # signal is described in greater detail below. Data output 259 of each channel in queue 204 is coupled to host processor data bus 24, for providing the stored bytes of received data to data bus 24, one word at a time. Output 259 also provides the corresponding 4-bit Byte En signal to host processor control bus 28.

During operation, byte to word group collection circuit 202 gets a byte of data from receive data queue 200 by activating the Rd RxQ/ signal. Byte to word group collection circuit 202 collects data bytes in queue 204 for all channels enabled by block 212. When all bytes of a word group have been collected in receive word group queue 204, byte to word group collection circuit 202 notifies buffer memory control circuit 104 over signal line 220, and buffer memory control circuit 104 writes the word group to buffer memory 20 by providing the appropriate address to multiplexer 208. The selected word group is presented at output 259 of receive word group queue 204 and written to buffer memory 20 over host processor data bus 24. While a word group is being transferred to buffer memory 20, byte to word group collection circuit 202 continues to collect data bytes for all enabled channels.

Queue # lookup table 210 provides parameters for the received data frames. In one embodiment, the lookup table

is essentially a dual port Random Access Memory (RAM) which is initialized through a first port at data input 222 and address input 224 by host processor 18 (shown in FIG. 5). Data output 226 and address input 228 form the second port. Address input 228 indexes the stored parameters to be presented at output 226 by using the incoming channel number Chan # from receive data queue 200. Each storage location within queue # lookup table 210 contains up to 32 bits of parameter information per channel number.

FIG. 7 is diagram which illustrates the bit definitions for each entry within queue # lookup table 210. Three parameters are defined per channel number. Bits 0-7, identified as "Queue #", specify which of the receive queues in queue 204 the received data frame is to be routed to. In one embodiment, the upper three bits of this field are coded as zeros. Bits 8-15, identified "Hdr Offset", define an integer offset which locates the beginning of the four byte frame header field embedded in the received data frame. If the first byte of the data frame is the beginning of the frame header field, then Hdr Offset is set to 0. Bits 16-23, identified as "Mask", are used to mask selected bits of the frame header field provided to receive frame action table 106. Each bit in the mask field can mask one bit or a group of bits in the frame header during a frame header lookup. Bits 24-31 are not defined in this embodiment and are coded as zeros.

Referring back to FIG. 6B, receive frame header array 206 is a small memory array which is used for storing the frame headers of received data frames. For each data frame received, buffer memory control circuit 104 and byte to word group collection circuit 202 capture the four byte (32-bit) frame header field, one byte at a time, in receive frame header array 206. When the byte of data presented at output 254 of receive data queue 200 corresponds to a byte of the frame header field, as identified by the Hdr Offset signal, that byte of data is stored in receive frame header array 206. The address at which the data is stored is provided at input 240 by byte to word group collection circuit 202. When all four bytes of the frame header have been captured, receive frame header array 206 routes the frame header field to output 242, which is coupled to comparison input 244 of receive frame action table 106.

Portions of the frame header field can also be routed to transmit word group queue 154 (shown in FIG. 6A) through multiplexers 156a-156d. For example, the frame header field may include a sequence number that may be inserted into a response data frame that is being transmitted from response message table 108 in response to the received data frame. The sequence number is inserted at multiplexers 156a-156d.

Receive frame action table 106 is preferably a Contents Addressable Memory (CAM) array which identifies an action to be performed as a function of the information contained within the frame header field that is applied to comparison input 244. The frame header field is masked with the Mask parameter that is applied to Mask input 230. Comparison output 246 includes a status signal which indicates whether the data within the masked frame header field is found within table 106. If the data is found within table 106, an action field corresponding to the data is provided on comparison output 246. If the data is not found within table 106, table 106 activates the status signal to instruct buffer memory control circuit 104 to discard the incoming data frame or to inform host processor 18 of an unknown incoming data frame. The contents of receive frame action table 106 is initialized by host processor 18 through data input 248 and address input 250.

FIG. 8 is a diagram illustrating the format of each entry within receive frame action table 106. In one embodiment,

table 106 includes up to 64 entries with each entry having two words. The first word of each entry includes "Byte 0", "Byte 1", "Byte 2" and "Byte3", which define the four byte frame header field that is to be compared against a received frame header field. The second word defines the frame action that buffer memory control circuit 104 is to perform when a frame header field match is found for Bytes 0-3. The frame action field is then applied to comparison output 246, which is coupled to buffer memory control circuit 104.

A variety of actions can be defined in the frame action field. In one embodiment, bit 0 defines a discard frame action. When bit 0 is set, data bytes are discarded until the incoming data frame has been completely received and the EOF flag has been detected. Bit 1 defines a host interrupt action. When bit 1 is set, buffer memory control circuit 104 generates a host processor interrupt DMAC Intp/ after the incoming data frame has been completely received and written into buffer memory 20. Bit 2 defines a send response frame action, and bits 4-7 provide the associated response frame number. When bit 2 is set, buffer memory control circuit 104 provides the 4-bit response frame number to address input 123 of response message table 108, which determines which response frame will be transmitted from the table. The response frame can be transmitted with or without a corresponding sequence number.

Referring to FIG. 6A, response message table 108 stores predetermined response messages that can be transmitted in response to received data frames. Table 108 is essentially a dual port Fast Action RAM (FAR) which is initialized through a first port at data input 134 and address input 136 by host processor 18 (shown in FIG. 5). Data output 122 and address input 123 form the second port. Address input 123 indexes a response data frame within response message table 108 by using the response frame number provided by buffer memory control circuit 104. Response message table 108 can have any length depending on how many response frames are to be stored and the number of bytes within each frame.

FIG. 9 is diagram which illustrates the format of each entry within response message table 108. There can be any number bytes of response data in each response frame. A response frame number of 0x00 would access the data frame beginning at address 0x3000. Similarly, a response frame number of 0x05 would access the data frame beginning at address 0x3050.

Response frames can also be transmitted from response message table 108 by an immediate command executed by host processor 18 from buffer memory 20. When host processor 18 determines that a response frame should be sent, host processor 18 selects the appropriate response frame in table 108 by using an address provided in buffer memory 20 as the address for response message table 108.

Referring again to FIG. 6B, the four Error Flags received by receive data queue 200 are routed through byte to word group collection circuit to buffer memory control circuit 104. The Error Flags are valid only when the EOF flag is set. At the end of a received data frame, buffer memory control circuit 104 writes the Error Flags into a Flag field of a current buffer memory descriptor within buffer memory 20. The buffer memory descriptor is described in more detail below with reference to FIGS. 15, 16, 18 and 19. Data errors are reported in this manner to allow the communications system to process the errored frames by maintaining statistics and discarding errored frames.

The four Error Flags are referred to as "receive data" errors and include an Rx Break/Abort Character Received

error, an Async Framing Error, an Rx CRC Error and an Rx Parity Error, for a serial WAN controller. An Rx Break/Abort Character Received error indicates a short frame has been received due an abort or break within the received data. An Async Framing Error indicates the serial WAN controller received a stop bit that was a zero. An Rx CRC Error indicates that the calculated CRC remainder did not match the expected remainder. An Rx Parity Error indicates a parity error was detected on the received data. "Hardware errors" are reported using the DMAC Error/signal output of buffer memory control circuit 104. This signal goes active when a hardware error occurs. The specific error can be read from a DMAC status register within configuration and status register block 212.

The registers within configuration and status register block 212 are shown in FIGS. 10-13. These registers are normally written to or read from during initialization and after an interrupt or error has occurred. In one embodiment, all registers within block 212 are accessed in Big Endian format. FIG. 10 is a diagram which illustrates the contents of a DMAC configuration register 260. DMAC configuration register 260 is used to configure parameters that apply to DMA controller 16 and all multiplexed channels. Bit 4 allows host processor 18 to reset DMA controller 16 by first writing bit 4 to a "1" to assert the reset signal and then writing bit 4 back to a "0" to deactivate the reset signal. Bit 2 enables DMA controller 16 to perform two and four word bursts to access buffer memory 20. When bit 2 is enabled, DMA controller 16 will burst two and four words automatically when the buffer memory address is on an 8 and 16 byte boundary, respectively. When bit 2 is disabled, DMA controller 16 accesses buffer memory 20 one word at a time. Bit 1 enables DMA controller 16 to transmit data from buffer memory 20 to interface controller 14. Bit 0 enables DMA controller 16 to receive data from interface 16 and to transmit the data to buffer memory 20.

FIG. 11 is a diagram which illustrates the bit definitions of a channel enable register 262 within block 212. Channel enable register 262 allows the user to selectively enable individual channels in DMA controller 16. A bit value of "1" enables the channel number that corresponds to the bit position number while a bit value of "0" disables the channel. For example, setting bit four will enable channel number 4.

FIG. 12 is a diagram which illustrates the bit definitions of DMAC status register 264 within block 212. Bit 15 indicates whether the transmit buffer memory descriptor is not ready. Bit 15 goes active when DMA controller 16 has fetched a buffer memory descriptor (BMD) from buffer memory 20 and the valid bit in the BMD has not yet been set. When this bit gets set, the transmit section for the particular channel becomes inactive until re-enabled by host processor 18. Bits 12-8 identify the number of the channel that is not ready.

Bit 15 interrupts are only reported if a subsequent BMD (not the first BMD of a data frame) is not ready. After the last buffer of a data frame has been processed, DMA controller 16 will fetch the next BMD from the BMD list. If the valid flag in this next BMD is set, then DMA controller 16 will continue to transmit more data. If the valid flag is not set, DMA controller 16 will halt the transmit section for that channel by clearing a corresponding BMD pointer active flag in a BMD pointer register within block 212, as discussed with reference to FIG. 13.

Bit 14 indicates whether a receive buffer memory descriptor within buffer memory 20 is not ready. Bit 14 will go

active after DMA controller 16 has fetched a buffer memory descriptor from buffer memory 20 and the valid bit in the descriptor is not set. When this bit gets set, the receive section will become inactive until re-enabled by host processor 18. Bits 12-8 of the DMAC status register 264 identify the channel number that is not ready. Bits 3-7 identify hardware errors. Five hardware errors are reported through bits 3-7, including a transmit data queue overrun error which indicates that transmit data queue 150 was full when transmit data was written; a transmit word group queue overrun error which indicates that transmit word group queue 154 was full when transmit data was written; a receive data queue underrun which indicates that receive data queue 200 was empty when receive data was read; a receive word group queue underrun which indicates receive word group queue 204 was empty when receive data was read; and a parity error which indicates a parity error was detected on the data bus 24 during a transmit operation. Bits 1 and 0 indicate whether the transmit completion and receive completion interrupts are active.

The DMAC Error/signal at the output of buffer memory control circuit 104 goes active when bits 7, 6, 5, 4 or 3 in DMAC status register 264 go active. The DMAC Intp/signal at the output of buffer memory control circuit 104 goes active when bits 15, 14, 1 or 0 in DMAC status register 264 go active.

FIG. 13 is a diagram which illustrates the bit definitions of the buffer memory descriptor (BMD) pointer registers within block 212. BMD pointer registers 266 include one transmit BMD pointer and one receive BMD pointer for each channel number, which point to the addresses of the active transmit buffer memory descriptor and the active receive memory descriptor in buffer memory 20. These descriptors provide the transmit and receive parameters for the data presently being transmitted and received. The BMD pointers are updated by DMA controller 16.

Bit 0 of each pointer is the pointer active flag. This flag informs DMA controller 16 whether the channel is active and whether DMA controller 16 can access the buffer memory descriptor in buffer memory 20. The pointer active flag is set by host processor 18 when it has set up valid descriptors in buffer memory 20. The pointer active flag is cleared by DMA controller 16 when it finds a descriptor that is not valid.

FIG. 14 is a diagram which illustrates the data structure in buffer memory 20 for a sequential BMD list. Buffer memory 20 includes sequential BMD list 270 and a plurality of data buffer locations 272. BMD list 270 includes a plurality of buffer memory descriptors 274. Each descriptor 274 includes several fields, including a byte count field 276, an immediate command field 278, a BMD flags field 280 and a buffer memory address field 282. There is one transmit BMD list 270 and one receive BMD list 270 for each channel. Each BMD pointer register 266 points to the presently active descriptor 274 within the corresponding BMD list 270. When the BMD pointer reaches the end of a BMD list, the BMD pointer is wrapped back around to point to the beginning of the BMD list.

FIG. 15 is diagram which illustrates the format of each descriptor 274 within BMD list 270 in greater detail. Each descriptor 274 within BMD list 270 is eight bytes long and is divided into two words, Word 0 and Word 1. Bits 31-16 of Word 0 are the byte count field 276 for data buffer 272. On transmit operations, this field specifies the number of bytes that are to be transferred from data buffer 272. On receive operations, this field is used for two different pur-

poses. When host processor 18 first sets up BMD list 270, this field defines how many bytes can be written into data buffer 272 (number of available bytes). After DMA controller 16 transfers the data into buffer 272, DMA controller 16 overwrites this field with the actual number of bytes that were written into buffer 272.

Bits 15-8 of Word 0 are the immediate command field 278, which is shown in more detail in FIG. 16. Bits 7-0 of Word 0 are the BMD flags field 280, which is shown in more detail in FIG. 16.

Bits 31-0 of Word 1 are the buffer memory address field 282 of data buffer 272, which points to the address of the first data byte of the data within data buffer 272.

FIG. 16 illustrates the bit definitions of immediate command field 278 and flags field 280. Bit 8 of Word 0 in the immediate command field indicates whether a response frame should be sent. When this bit is a 1, DMA controller 16 transmits a data frame from response message table 108. Byte count field 276 defines the number of bytes to send. Buffer address field 282 (shown in FIG. 15) gives the starting address in response message table 108 from which to transmit the data.

Bits 7-4 of Word 0 in flags field 280 are the four error flags that are received from receive data queue 200. These flags are valid when the end of frame (EOF) flag, bit 1 of Word 0, is also set. At the end of a received data frame, DMA controller 16 writes these flags to the flags field 280 of the current buffer memory descriptor 274.

Bit 3 of Word 0 defines a buffer completion interrupt, which allows DMA controller 16 to activate the DMAC Intp/signal when the data transfer has taken place for the current descriptor 274. If this bit is "0", no interrupt signal will activate and DMA controller 16 will continue to the next descriptor 274 in BMD list 270.

Bit 2 of Word 0 identifies the start of a frame (SOF) and is a "1" when descriptor 274 contains the parameters for the first buffer of a data frame. Bit 1 of Word 0 identifies the end of a frame (EOF), and is a "1" when descriptor 274 contains the parameters for the last buffer of a data frame. If this bit is a "0", the end of a data frame has not been reached and DMA controller 16 will continue to the next descriptor 274 in BMD list 270. The entire data frame is contained in one data buffer 272 when both the start of frame and end of frame flags are set in one descriptor 274.

Bit 0 of Word 1 is a valid parameter, which is a "1" when the descriptor contains valid parameters for a data buffer and a data transfer can take place with buffer memory 20. If this bit is a "0", then a data transfer with buffer memory 20 cannot take place. BMD pointer active flag, bit 0 of BMD pointer register 266, will be cleared by DMA controller 16 and the DMAC Intp/signal will be activated.

FIG. 17 is a diagram which illustrates the data structure in buffer memory 20 for a linked BMD list 290 according to an alternative embodiment of the present invention. Linked BMD list 290 includes a set of link accessed buffer memory descriptors 292, which point to data within a data buffer 294. BMD pointer register 266 points to the first descriptor 292 in linked list 290. Each descriptor 292 includes a byte count field 300, an immediate command field 302, a BMD flags field 304, a buffer address field 306 and a next BMD pointer field 308. Byte count field 300, immediate command field 302, BMD flags field 304 and buffer address field 306 are substantially the same as the corresponding fields shown in FIGS. 15 and 16 and are used in a similar manner. The next BMD pointer field 308 points to the next descriptor 292 in linked list 290. When DMA controller 16 reaches the end of

a BMD list 290, the next BMD pointer field 308 of the last descriptor 292 is equal to "0".

When DMA controller 16 removes one or more descriptors 292 from linked list 290, it sets its current BMD pointer 266 equal to the next BMD pointer 308 of the current descriptor 292. When host processor 18 adds one or more descriptors 292 to the end of linked list 290, it sets the next BMD pointer field 308 of the last descriptor 292 equal to the buffer memory address of the first descriptor 292 that is being added. Host processor 18 can also add one or more descriptors to the middle of a linked list 290 by modifying the next BMD pointer field 308 of a descriptor 292 to insert the subsequent link and by modifying the next BMD pointer field 308 of the last added descriptor 292 to complete the link operation.

FIGS. 18 and 19 are diagrams which illustrate the format of a descriptor 292 in greater detail. Each descriptor 292 is 16 bytes long and is divided into four words. Bits 31-16 of Word 0 contain byte count field 300. Bits 15-8 Word 0 contain intermediate command field 302. Bits 7-0 of Word 0 contain BMD flags field 304. Bits 31-0 of Word 1 contain buffer memory address field 306. Bits 31-0 of Word 2 contain next BMD pointer field 308. This is the address in buffer memory 20 of the next descriptor 292 in linked list 290. Since a descriptor 292 starts on a 16 byte address boundary, bits 3-0 of next BMD pointer field 308 are expected to be zero. Bits 31-0 of Word 3 are not used.

CONCLUSION

The DMA controller of the present invention improves efficiency the host processor and the host processor data bus in several ways. First, the DMA controller has enhanced data frame processing capability which relieves the host processor from much of the burden of deciding what to do with an incoming data frame. Predetermined commands are now stored in the receive frame action table within the DMA controller. One of the commands stored in the receive frame action table is to automatically create a response frame. If a response frame needs to be sent as an acknowledgment to an incoming data frame, the host processor does not have to build the response frame. An appropriate, predetermined response frame is already residing in a response message table within the DMA controller. Also, the host processor does not need to notify the DMA controller to transfer the response frame. The receive frame action table automatically notifies the buffer memory control circuit when to send a response frame from the response message table. Finally, the DMA controller does not need to use the host processor data bus to send the response frame. Since the response frame is transferred from the response message table, which is part of the DMA controller, the DMA controller does not need to access the host processor data bus. The host processor maintains control over all aspects of data frame decision making and response frame generation since the queue # lookup table, the receive frame action table and the response message table are all programmable by the host processor.

Although the present invention has been described with reference to preferred embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention. For example, the number of channels, bytes, and bits used in the various elements of the DMA controller of the present invention are arbitrary and can be varied as desired in alternative embodiments. Also, the particular data formats and structures of the various tables and queues can

be varied in alternative embodiments. The term "coupled" used in the specification and the claims includes various types of connections or couplings and can include a direct connection or a connection through one or more intermediate components.

What is claimed is:

1. A direct memory access (DMA) controller for transmitting and receiving formatted data frames having frame headers in a communications subsystem, the DMA controller comprising:

a transmit data input and a transmit data output; transmit circuitry receiving transmit data frames on the transmit data input and applying the transmit data frames on the transmit data output through a transmit data path;

a receive data input and a receive data output; receive circuitry receiving receive data frames on the receive data input and applying the receive data frames on the receive data output through a receive data path; and

a response message table having an address input coupled to the receive circuitry, a data output coupled to the transmit data path of the transmit circuitry and a plurality of addressable memory locations for storing predetermined response data frames associated with particular ones of the receive data frames.

2. The DMA controller of claim 1 and further comprising:

a receive frame header capture circuit having a data input coupled to the receive data circuitry for capturing the frame header of each receive data frame and having a data output; and

a receive frame action table having an input coupled to the data output of the receive frame header capture circuit, a plurality of addressable memory locations for storing predetermined action commands, and an action command output which is coupled to the address input of the response message table.

3. The DMA controller of claim 2 wherein the receive frame header capture circuit comprises a memory array.

4. The DMA controller of claim 2 wherein the receive frame action table comprises:

a contents addressable memory (CAM) having a comparison input coupled to the data output of the receive frame header capture circuit and a comparison output provided to the address input of the response message table, wherein each addressable memory location in the CAM includes a frame header field and an associated frame action field and wherein the associated frame action field is applied to the comparison output if the captured frame header corresponds to the frame header field of that memory location.

5. The DMA controller of claim 4 wherein the associated frame action field comprises a send response frame action field which identifies whether a response data frame shall be transmitted from the response message table and a response frame number field which identifies an address in the response message table at which the response data frame is stored.

6. The DMA controller of claim 5 wherein the associated action field further comprises a discard frame action field and a host processor interrupt action field.

7. The DMA controller of claim 1 wherein the response message table comprises a random access memory.

8. A direct memory access (DMA) controller for transmitting and receiving formatted data frames having frame headers in a communications subsystem, the DMA controller comprising:

15

a transmit data input and a transmit data output;
transmit circuitry receiving transmit data frames on the
transmit data input and applying the transmit data
frames on the transmit data output;

a receive data input and a receive data output;
receive circuitry receiving receive data frames on the
receive data input and applying the receive data frames
on the receive data output;

a receive frame action table having an input coupled to the
receive circuitry, a plurality of memory locations for
storing predetermined action commands which are
addressed by the frame headers, and an action com-
mand output;

a receive frame header capture circuit having a data input
coupled to the receive circuitry for capturing the frame
header of each receive data frame and having a data
output coupled to the input of the receive frame action
table; and

wherein the receive frame action table comprises a con-
tents addressable memory (CAM) having a comparison
input coupled to the data output of the receive frame
header capture circuit and a comparison output which
forms the action command output, wherein each
addressable memory location in the CAM includes a
frame header field and an associated frame action field
and wherein the associated frame action field is applied
to the comparison output if the captured frame header
corresponds to the frame header field of that memory
location.

9. The DMA controller of claim 8 and further comprising:

a response message table having an address input coupled
to the comparison output, a data output coupled to the
transmit circuitry and a plurality of addressable
memory locations for storing predetermined response
data frames.

10. The DMA controller of claim 9 wherein the associated
frame action field comprises a send response frame action
field which identifies whether a response data frame shall be
transmitted from the response message table and a response
frame number field which identifies an address in the
response message table at which the response data frame is
stored.

11. The DMA controller of claim 9 wherein the associated
frame action field further comprises a discard frame action
field and a host processor interrupt action field.

12. A communications subsystem for transmitting and
receiving formatted data frames having frame headers, the
communications subsystem comprising:

a communications interface;

a host processor interface;

a host processor and a buffer memory coupled to the host
processor interface; and

a direct memory access (DMA) controller coupled
between the communications interface and the host
processor interface, the DMA controller comprising:

a transmit circuit which receives transmit data frames
on the host processor interface and applies the trans-
mit data frames to the communications interface;

a receive circuit which receives receive data frames on
the communications interface and applies the receive
data frames to the host processor interface;

a receive frame action table having a frame header
input coupled to the receive circuit, a frame action
output and a plurality of frame action memory loca-
tions which are addressed by the frame header input;
and

16

a response message table having a first address input
coupled to the frame action output, a response mes-
sage output coupled to the transmit circuit and a
plurality of response message memory locations
which are addressed by the frame action output.

13. The communications subsystem of claim 12 wherein
each frame action memory location comprises:

a send response frame field which identifies whether a
response frame shall be transmitted from the response
message table in response to the receive data frame;
and

a response frame identifier field which identifies the
response memory location from which the response
frame will be transmitted.

14. The communications subsystem of claim 12 and
further comprising:

a receive frame header capture circuit having a data input
coupled to the receive circuitry for capturing the frame
header of each receive data frame and having a data
output coupled to the frame header input of the receive
frame action table.

15. The communications subsystem of claim 12 wherein
the receive frame action table comprises:

a contents addressable memory (CAM) having a com-
parison input coupled to the data output of the receive
frame header capture circuit and a comparison output
which forms the frame action output, wherein each
frame action memory location in the CAM includes a
send response frame action field, a response frame
identifier field and a frame header field which is com-
pared with the frame header applied to the frame header
input.

16. The communications subsystem of claim 15 wherein
each frame action memory location further comprises a
discard frame action field and a host processor interrupt
action field.

17. The communications subsystem of claim 15 wherein
the CAM further comprises a data input and an address input
which are coupled to the host processor interface.

18. The communications subsystem of claim 15 wherein
the DMA controller further comprises:

a lookup table having first and second data ports, first and
second address inputs and a plurality of table entries
with a mask field which is applied to the first data port
when addressed by the first address input, wherein the
first address input is coupled to the receive circuit and
the second address input and second data port are
coupled to the host processor interface; and

wherein the CAM further comprises a mask input coupled
to the first data port, which masks the frame header
field applied to the frame header input.

19. The communications subsystem of claim 12 wherein
the response message table comprises a dual port random
access memory having a first port which includes the first
address input and the response message output and a second
port which includes a second address input and a data input
which are coupled to the host processor interface.

20. The communications subsystem of claim 12 wherein
the buffer memory comprises:

a list of buffer memory descriptors, each descriptor having
a byte count field, an immediate command field and a
buffer memory address field, wherein the immediate
command field comprises a send response frame field
which identifies whether a response frame shall be
transmitted from the response message table.

21. A communications subsystem for transmitting and
receiving formatted data frames having frame headers, the
communications subsystem comprising:

17

a communications interface;
 a host processor interface;
 a host processor and a buffer memory coupled to the host processor interface; and
 a direct memory access (DMA) controller coupled between the communications interface and the host processor interface, the DMA controller comprising:
 transmit means for receiving transmit data frames on the host processor interface and applying the transmit data frames to the communications interface;
 receive means for receiving receive data frames on the communications interface and applying the receive data frames to the host processor interface;
 frame action determination means for determining an action to be performed on each receive data frame as a function of the frame header; and
 response message generator means for generating a response frame and applying the response frame to the transmit means as a function of the action determined by the frame action determination means.

22. The communications subsystem of claim 21 wherein the response generator means comprises means for storing a plurality of predetermined response frames which are addressed by the action determined by the frame action determination means.

23. The communications subsystem of claim 21 wherein the buffer memory comprises means for initiating transmission of a selected predetermined response frame stored in the response message generator means by the transmit means.

24. The communications subsystem of claim 21 wherein the frame action determination means comprises means for storing predetermined frame header fields and associated

18

frame action fields which are addressed by the frame headers of the receive data frames.

25. The communications subsystem of claim 24 wherein the DMA controller further comprises:

means for capturing the frame header of each receive data frame and applying the frame header to the frame action determination means.

26. A method of processing a formatted data frame having a frame header in a communications subsystem which comprises a communications interface, a DMA controller and a host processor interface, the method comprising:

receiving the data frame within the DMA controller from the communications interface;

capturing the frame header from the received data frame within the DMA controller;

applying the received data frame to the host processor interface;

applying the frame header to a comparison input of a contents addressable memory (CAM) which has a plurality of addressable memory locations, each addressable memory location having a frame header field and an associated frame action field and wherein the associated frame action field is applied to a comparison output of the CAM if the frame header corresponds to the frame header field;

generating a predetermined response frame within the DMA controller as a function of the associated frame action field applied to the comparison output; and

transmitting the response frame from the DMA controller to the communications interface.

* * * * *



US006282454B1

(12) **United States Patent**
Papadopoulos et al.

(10) **Patent No.:** **US 6,282,454 B1**

(45) **Date of Patent:** **Aug. 28, 2001**

(54) **WEB INTERFACE TO A PROGRAMMABLE CONTROLLER**

(75) **Inventors:** **A. Dean Papadopoulos**, Groton; **Allan Tanzman**, Newton Center; **Richard A. Baker, Jr.**, West Newbury; **Rodolfo G. Bellardi**, Malden, all of MA (US); **Dennis J. W. Dube**, Pelham, NH (US)

(73) **Assignee:** **Schneider Automation Inc.**, North Andover, MA (US)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) **Appl. No.:** **08/927,005**

(22) **Filed:** **Sep. 10, 1997**

(51) **Int. Cl.⁷** **G05B 9/02**

(52) **U.S. Cl.** **700/83; 700/67**

(58) **Field of Search** **700/67, 83; 707/10, 707/2; 345/335**

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,971,000 7/1976 Cromwell .

4,319,338 3/1982 Grudowski et al. .

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

296 00 609

U1 3/1997 (DE) .

441 0 171 C1 4/1997 (DE) .

196 15 093

A1 10/1997 (DE) .

0 542 657 A1 5/1993 (EP) .

0 814 393 A1 12/1997 (EP) .

WO 97/18636 5/1997 (WO) .

WO 98/53581 11/1998 (WO) .

(List continued on next page.)

OTHER PUBLICATIONS

http://www.adeptsience.com/archive_pressroom/html/la-btechnet.html; Adapt PressRoom Archives. A collection of Adept Scientific's archive news releases. Hot Coffee on the Internet!.

(List continued on next page.)

Primary Examiner—William Grant

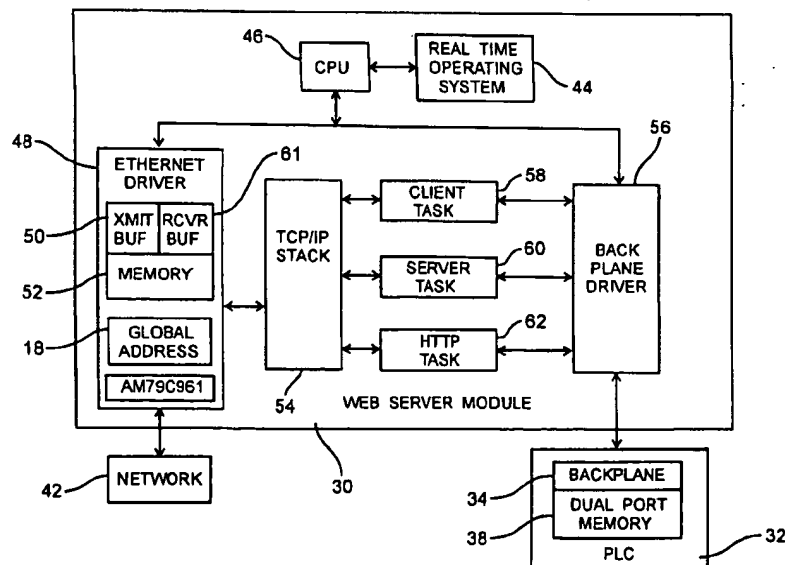
Assistant Examiner—Victoria Robinson

(74) **Attorney, Agent, or Firm**—Michael J. Femal; Larry I. Golden

(57) **ABSTRACT**

A control system includes an Internet web interface to a network of at least one programmable logic control system running an application program for controlling output devices in response to status of input devices. The Web interface runs Web pages from an Ethernet board coupled directly to the PLC back plane and includes an HTTP protocol interpreter, a PLC back plane driver, a TCP/IP stack, and an Ethernet board kernel. The Web interface provides access to the PLC back plane by a user at a remote location through the Internet. The interface translates the industry standard Ethernet, TCP/IP and HTTP protocols used on the Internet into data recognizable to the PLC. Using this interface, the user can retrieve all pertinent data regarding the operation of the programmable logic controller system.

18 Claims, 4 Drawing Sheets



U.S. PATENT DOCUMENTS

4,688,167	8/1987	Agarwal .	
4,845,644	7/1989	Anthias et al. .	
4,858,152	8/1989	Estes .	
4,897,777	1/1990	Janke et al. .	
4,912,623	3/1990	Rantala et al. .	
4,937,777	6/1990	Flood et al. .	
4,949,274	8/1990	Hollander et al. .	
4,953,074	8/1990	Kamelani et al. .	
4,992,926	2/1991	Janke et al. .	
5,012,402	4/1991	Akiyama .	
5,023,770	6/1991	Siverling .	
5,047,959	9/1991	Phillips et al. .	
5,072,356	12/1991	Watt et al. .	
5,072,412	12/1991	Henderson, Jr. et al. .	
5,109,487	4/1992	Ohgomori et al. .	
5,122,948	6/1992	Zapolin .	
5,131,092	7/1992	Sackmann et al. .	
5,134,574	7/1992	Beaverstock et al. .	
5,151,896 *	9/1992	Bowman et al.	370/85.13
5,151,978	9/1992	Bronikowski .	
5,157,595	10/1992	Lovrenich .	
5,159,673	10/1992	Sackmann et al. .	
5,161,211	11/1992	Taguchi et al. .	
5,165,030	11/1992	Barker .	
5,179,700	1/1993	Aihara et al. .	
5,225,974	7/1993	Mathews et al. .	
5,245,704	9/1993	Weber et al. .	
5,251,302 *	10/1993	Weigl et al.	395/250
5,283,861	2/1994	Dangler et al. .	
5,297,257	3/1994	Struger et al. .	
5,307,463	4/1994	Hyatt et al. .	
5,321,829	6/1994	Zifferer .	
5,349,675	9/1994	Fitzgerald et al. .	
5,398,336	3/1995	Tantry et al. .	
5,406,473	4/1995	Yoshikura et al. .	
5,420,977	5/1995	Sztipanovits et al. .	
5,440,699	8/1995	Farrand et al. .	
5,446,868	8/1995	Gardea et al. .	
5,528,503	6/1996	Moore et al. .	
5,598,536	1/1997	Slaughter, III et al. .	
5,613,115	3/1997	Gihl et al. .	
5,623,652	4/1997	Vora et al. .	
5,625,781	4/1997	Cline et al. .	
5,699,350 *	12/1997	Kraslavsky	370/254
5,734,831 *	3/1998	Sanders	395/200.53
5,805,442 *	9/1998	Crater et al.	364/138
5,950,006	9/1999	Crater et al. .	
5,975,737	11/1999	Crater et al. .	
5,982,362	11/1999	Crater et al. .	
5,997,167	12/1999	Crater et al. .	

OTHER PUBLICATIONS

Using World-Wide Web for Control Systems,, F. Momal, C. Pinto-Pereira, AT Division CERN, 1211 Geneva 23, <http://mish231.cern.ch/Docs/ICALEPCS/1995/icalp95.htm>.

"Networking Reference Manual," 1994 National Instruments Corporation, Part No. 320587B-01, Sep. 1994, Copyright © 1993.

"Data Acquisition VI Reference Manual for Windows," 1994 National Instruments Corporation, Part No. 320536B-01, Sep. 1994, Copyright © 1992.

"Tutorial for Windows," 1994 National Instruments Corporation, Part No. 320593B-01, Sep. 1994, Copyright © 1993.

Rockwell International Corporation—Allen-Bradley—Networks—Ethernet for Industrial Control—An Ethernet White Paper—Apr. 21, 1998, pp. 1-13.*

Rockwell International Corporation—Automation Systems Control—General—World-Class Automation Systems from Allen-Bradley, Last Updated: May 7, 1998, pp. 1-12.*

PC QUEST, Dec. '97—Point, click, Control—C—Programmable controllers take the pain out of embedded control, pp. 1-2.*

berthel—automation with imagination—PCI 100—Programmable logic controller for SIMATIC/IBM IPC, pp. 1-3.*

Yahoo! Personalized Search Results for programmable logic controller internet access, pp. 1-3.*

Siemens—SIMATIC report 1/97—New in the SIMATIC Library, pp. 1-2.*

Control Magazine Aug. 1998—Field Test—Dynamic Software Makes Control Integration Easier, pp. 1-2.*

Design and Reuse Web Site—EDTN Network—Analyze IP Database Content—Analyze Reuse Blocks per taxonomy tree, pp. 1-10.*

Engineering Information, Inc.—Ei CPX WEB [1990-94].* Using World Wide Web for Control Systems, F. Momal, C. Pinto-Pereira, AT Division CERN, 1211 Geneva 23, <http://mish231.cern.ch/Docs/ICALEPCS/1995/icalp95.htm>.*

"Ethernet Base Gateway Product," AEG-Modicon, published 1991.*

"Modicon Modbus Plus Network BM85 Bridge Multiplexer User's Guide," Groupe Schneider, Aug. 1995.*

"Modicon Modbus Plus Network Planning and Installation Guide," AEG Schneider Automation, Apr. 1996.*

"Open Modbus/TCP Specification," A. Swales, Sep. 3, 1997.*

"MEB Installation and Programming Manual," Niobrara Research and Development Corporation, Sep. 24, 1997.*

"MEB-TCP Installation and Programming Manual," Niobrara Research and Development Corporation, Oct. 1, 1997.*

"Internet Protocol, Darpa Internet Program, Protocol Specification—RFC:791," Defense Advanced Research Projects Agency, Sep. 1981.*

"Transmission Control Protocol, Darpa Internet Program, Protocol Specification—RFC:793," Defense Advanced Research Projects Agency, Sep. 1981.

"Open MODBUS/TCP Specification," A. Swales, Sep. 3, 1997.

"[comp.unix.programmer] Unix-Socket-FAQ For Network Programming," Vic Metcalfe, Andrew Gierth and other contributors, Jan. 22, 1998.

"TCP/IP Illustrated, vol. 2, The Implementation," Gary R. Wright, W. Richard Stevens, 1997.

"Winsock 2 Information," Bob Quinn, 1995-1998 (last updated Dec. 5, 1998).

Website Information of PROFIBUS: Technical Overview.

Website Information of ODVA—The Open DeviceNet's Vendor Association.

Website of PROFIBUS International—Welcome Page.

When Technology Standards Become Counterproductive, Kenneth C. Crater, President, Control Technology Corporation, Hopkinton, MA dated Jul. 9, 1999, pp. 1-5.

A White Paper State Language for Machine Control, Kenneth C. Crater, President, Control Technology Corporation, Hopkinton, MA dated Jul. 9, 1999, pp. 1-11.

New PC-based Process Control & Data Acquisition Software Integrates Remote Internet Capabilities with Fast Pentium Support, Fred A. Putnam, Labtech President, pp. 1-3.

- Aug. 1996 Control Magazine—In The News—Electric Utility Industry Embarks on Automation Overhaul, pp. 1–10.
- Jul. 1997 Control Magazine—Magazine Software Review—NT Package Give Plant Access Through the Web, pp. 1–3.
- Oct. 1996 Control Magazine—Software Review—Article Archives, pp. 1–2.
- ICS Instrumentation & Control Systems—Windows NT for real-time control: Which way to go?—ICS Magazine, pp. 1–8.
- I&CS Jul. 1999—Special Report Software—Software: Open source OSs, objects, Web-based communications challenge status quo, (Wayne Labs, Senior Technical Editor), pp. 24–49.
- Landis & Staefa MS 2000, pp. 1–2.
- Landis & Staefa Standards and Open Protocols Integration System Architecture, p. 1.
- Annabooks Bookstore, Programming and Interfacing the 8051, by Sencer Yeralan and Asutosh Ahluwalia, pp. 1–2.
- SoftPLC Corporation—Java Support in SoftPLC Corp. Products, pp. 1–5.
- Mach J. Company, MachJ, an embeddable, clean room Java Virtual Machine, p. 1.
- SoftPLC Corporation—The History of Programmable Controllers, Looking Back From the Year 2000 A.D. (Or, How Computers Replaced Proprietary PLC'S), pp. 1–7.
- SoftPLC Corporation—TOPDOC: Advanced PLC program development & documentation software, pp. 1–12.
- Control Engineering Online Magazine Articles (Jul. 1998)—No. that's not a PC, it's a PLC, pp. 1–2.
- Rockwell International Corporation, Allen-Bradley Introduces PLC-5/80E Controller for Ethernet Communication Networks.
- Rockwell Automation—Search Results, pp. 1–2.
- Rockwell International Corporation, Vision & Direction, The Direction of Automation Systems, pp. 1–4.
- Rockwell International Corporation, Vision & Direction, The Role of Open Systems, pp. 1–4.
- Rockwell International Corporation—Vision & Direction—The Direction of Automation Systems—Emergence of Application-Specific Control Solutions, pp. 1–2.
- Rockwell International Corporation—Vision & Direction—The Direction of Automation Systems—The New Factory Worker, pp. 1–2.
- Rockwell International Corporation, Vision & Direction, Control System Deliverables—The Next Step, pp. 1–2.
- Rockwell International Corporation, Vision & Direction, Conclusion & Acknowledgments, pp. 1–2.
- Rockwell International Corporation—Choices—Perspectives on the Future of Automation Control, p. 1.
- U.S. Patent Application No. ble Controller; Inventor: Pap. Abstract of "Implementing distributed controls for FMC's using Internet utilities," S. S. Jagdale and N. Merchant; Computers of Industrial Engineering, vol. 31 No. 1–2, pp. 87–90; Oct., 1996 (UK).
- Abstract of "Process Control takes to the Net," Greg Paula, Mechanical Engineering vol. 118 No. 12 Dec. 1996, p. 55.
- Abstract of "Remote interrogation and control of sensors via the internet," Peter L. Furr and Euan F. Mowat; Sensors, vol. 12 No. 12, pp. 6; Dec. 1995.
- Abstract of "Process control takes to the Net," G. Paula; Mechanical Engineering, vol. 118, No. 12, p. 55, Dec., 1996.
- Abstract of Implementation of CAN/CAN bridges in distributed environments and performance analysis of bridged CAN systems using SAE benchmark, H. Ekiz, A. Kutlu and E. T. Powner; Conference Paper, IEEE Southeastern '97, Engineering the new energy, IEEE, p. 185–7, 1996.
- Abstract of "Managing interdisciplinary project teams through the Web," R. E. Goodman and P. Chinowsky; Conference Paper, WebbNet 96 –World Conference of the Web Society, pp. 180–5, 1996.
- Abstract of "Learning environment for a process automation system using computer networks," J. Lindfors, L. Yliniemi and K. Leivska; Conference Paper, Step '96 –Genes, Nets and Symbols, pp. 137–43, 1996 (Finland).
- Abstract of "Distributed agent systems for intelligent manufacturing," D. H. Norrie and B. R. Gaines; Canadian Artificial Intelligence, No. 40, p. 31–3, Autumn 1996 (Canada).
- Abstract of Proceedings of AUTOFACT 1995 Conference, "Today's Automated, Integrated Factory," Soc. Manuf., Eng., Dearborn, MI; 1995.
- Abstract of "The ECOSSE Control HyperCourse," C. M. Merrick and J. W. Ponton; Computers & Chemical Engineering, vol. 20, Part B, p. S 1353–8, 1996 (UK).
- Abstract of "Chemical-better batch controls," T. Crowl; Control & Instrumentation, vol. 28, No. 5, p. 53–4, May 1996 (UK).
- Abstract of "Industrial Software does 32-bit Windows, prepares for the net," W. Labs; I 8CS, vol. 69, No. 3, p. 23–6, 31–4, Mar. 1996, USA.
- Abstract of "A Case study for international remote machining," G. C. I. Lin and Kao Yung-Chou; Conference Paper, Proc. SPIE-Int. Soc. Opt. Eng., vol. 2620, pp. 553–60, 1995.
- Abstract of "Standardization of long-distance protocols," R. Dinges; Journal Paper, Generation Changes in Network Conductor Systems, ITG-Fachberichte, vol. 134, pp. 97–113, 1995 (West Germany).
- Abstract of "Proceedings of AUTOFACT Conference," Soc. Manuf. Eng., , p. 684, Dearborn MI; 1993.
- Abstract of "Control system design V. Communications orchestrate process control," F. Glow; In Tech, vol. 36, No. 9, pp. 68–74, Sep. 1989.
- Abstract of "Functions and characteristics of local networks adapted to industrial applications," J. Morlais; Electronique Industrielle, No. 97, pp. 56–63, Nov. 15, 1985; France.
- Abstract of "Intelligent supervisory control of submerged-arc furnaces," Markus A. Reuter, Carla Pretorius, Chloe West, Pater Dixon and Mome Oosthuizen, JOM vol. 48, No. 12, Dec. 1996, pp. 49–51.
- Abstract of "Simulation on the integration of process control systems of rolling mill plants through standard networks," Choo Young Yeol, Hwang Hwa Won and Kim Checha, Proceedings of the Industrial Computing Conference, Instrument Society of America, Research Triangle Park, NC, USA. pp. 1–14; vol. 6, No. 1, 1996.
- Abstract of "Environmental waste control digest," Clayton H. Billings; Public Works vol. 127 No. 7, p. 6, Jun., 1996.
- Abstract of "Experiments in tele-handling and tele-machining at the macro and micro scales, using the Internet for operational environment transmission," Mamoru Mitsuishi, Toshio Hori, Tomoharu Hikita, Masao Teratani, Takuro Watanabe, Hirofumi Nakanishi and Bruce Kramer; IEEE International Conference on Intelligent Robots and Systems vol. 2, 1995.

Abstract of "A phototyping and reverse engineering system for mechanical parts-on-demand on the national network," Fred Hansen, Elias Pavlakos, Eric Hoffman, Takeo Kanade, Raj Reddy, Paul Wright; Journal of Manufacturing Systems, vol. 12 No. 4, pp. 269-281; 1993.

Abstract of "Mathematical model and optimization of furfural treating process," Tao Pheng, Jinshou Yu and Huihe Shao; Huadong Huagong Xueyuan Xuebao/Journal of East China Institute of Chemical Technology vol. 17 No. 1, pp. 99-104; Feb. 1991.

Abstract of User's Aspect of Telecommunication and Information Processing in Plant Factory; Hashimoto Yasushi (1); Journal of the Institute of Electronics, Information and Communication Engineers, vol. 78, No. 5, pp. 475-81, Fig. 3, Ref. 7, 1995. (Japan).

Abstract of "High-efficient application technology of DCS from the viewpoint of users," Oka Norihito (1); Narita Tsutomu (1); (1) Yamatake-Honeywell Co., Ltd.; Otomeshon, vol. 40, No. 2, pp. 24-28, Fig. 5, Part 2, 1995. (Japan).

Abstract of Users' experience with software tools for process integration. General results; Stougie, L.; Roeterink, H.J.H.; Van Wijk, A.; Stikkelman, R.M.; Nov. 1996.

Abstract of "Integrated design and process technology. vol. 1;" Tanik, M.M.; Bastani, F.B.; Sheu, P.C.Y.; Tsai, J.P.; Mittermeir, R.; Society for Design and Process Science, pp. 51-57; 1996. (USA).

Abstract of "Integrated design and process technology. vol. 2;" Tanik, M.M.; Bastani, F.B.; Gibson, D.; Felding, P.J.; Society for Design and Process Science, pp. 423-430, 1996. (USA).

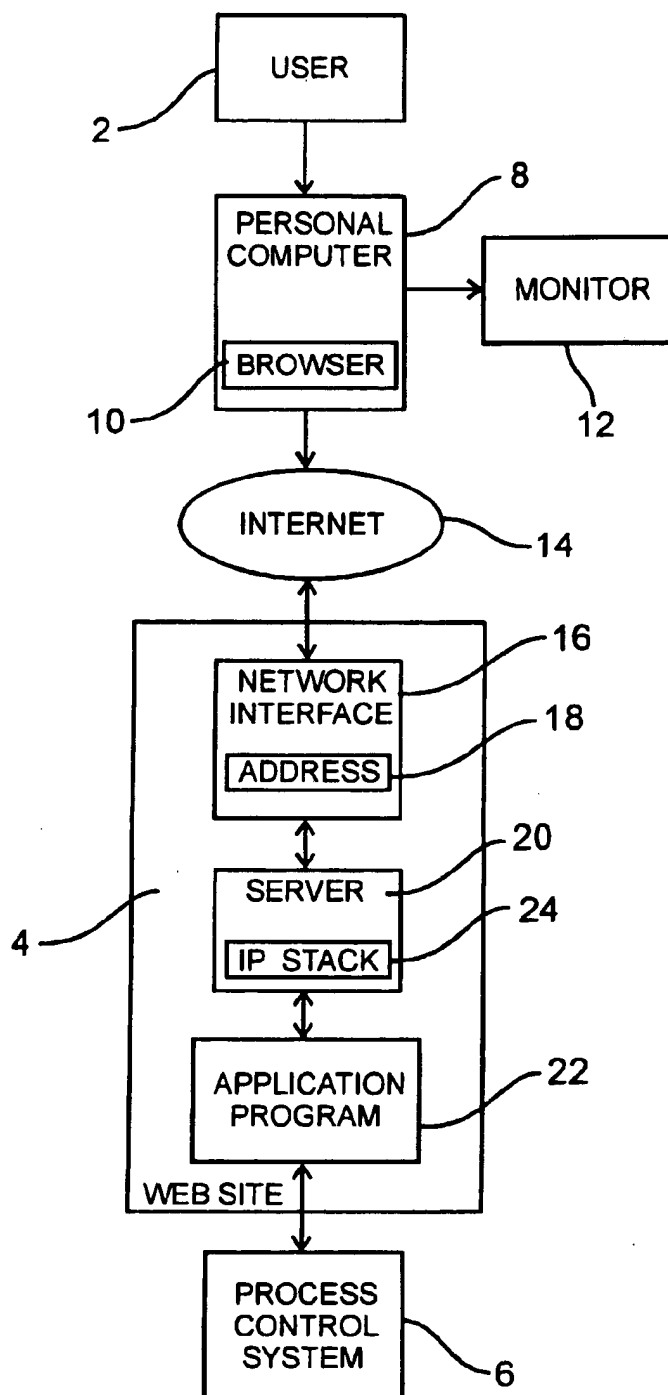
Abstract of "Integrated design and process technology. vol. 2" Tanik, M.M.; Bastani, F.B.; Gibson, D.; Fielding, P.J.; Society for Design and Process Science, pp. 306-312, 1996.

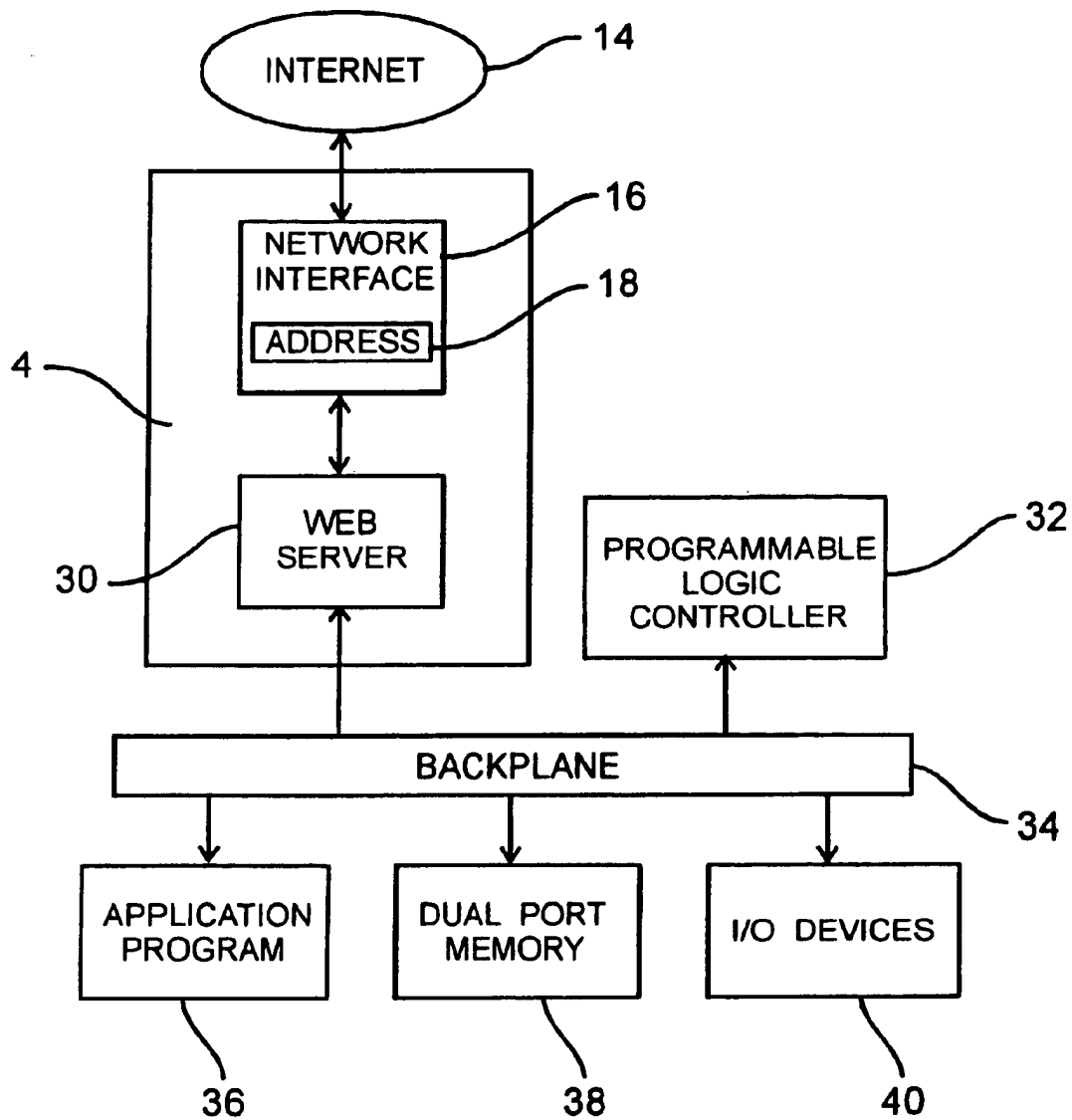
Abstract of "Need low-cost networking consider DeviceNet," W. H. moss; InTech vol. 43:11; pp. 30-31, Nov. 1996.

"Plastic Car Bodies Pass the Crash Test," mechanical engineering; vol. 118, No. 12; Dec. 1996.

"Remote Interrogation and Control of Sensors via the Internet," Sensors and Systems; Peter L. Fuhr and Euan F. Mowat; University of Vermont; pp. 25-30; Dec. 1999.

* cited by examiner

*Fig. 1*

*Fig. 2*

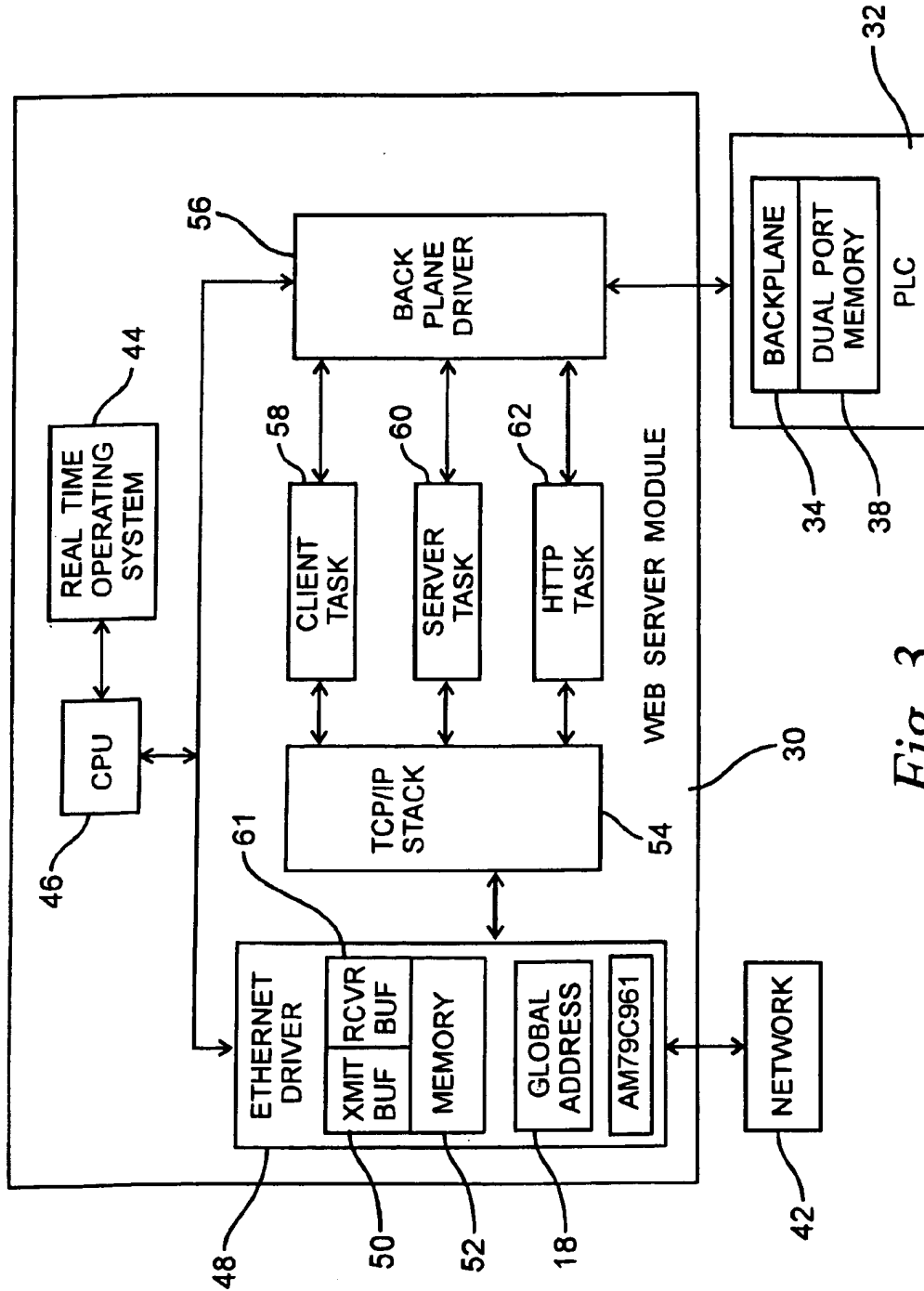
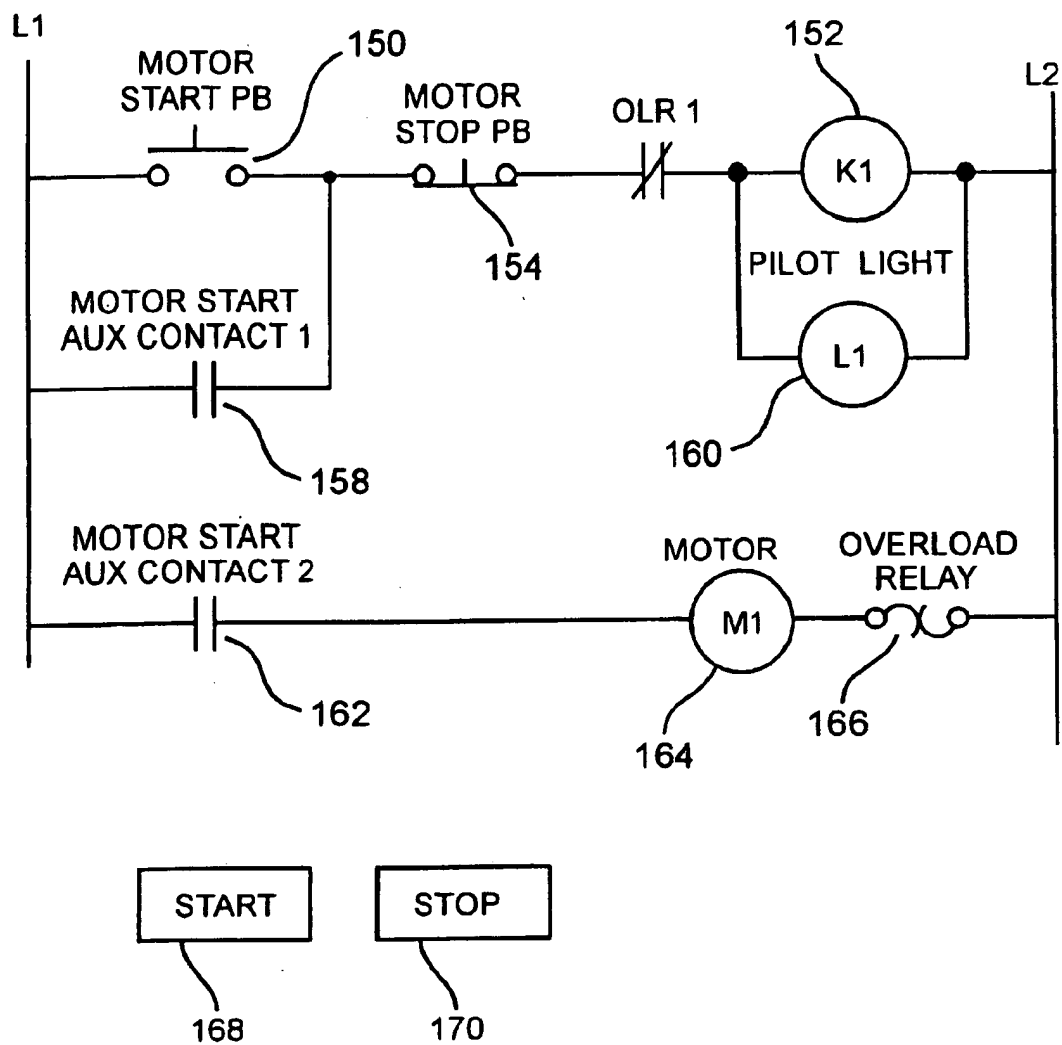


Fig. 3

*Fig. 4*

1

WEB INTERFACE TO A PROGRAMMABLE CONTROLLER

TECHNICAL FIELD

Applicants' invention relates generally to the field of programmable controllers and more particularly to a system for coupling a network of programmable controllers through an internetwork to a monitoring and control device.

RELATED APPLICATIONS

This application is related to the following, commonly assigned application filed concurrently herewith, entitled "Apparatus for Controlling Internetwork Communications" (Application Ser. No. 08/926,837). The contents of these Applications are expressly incorporated herein by reference.

BACKGROUND ART

Remote monitoring and control of systems and processes have taken many forms. In the past, dedicated lines became the most common form of communication between a control system and a remote location. This has limited application since the control system was not accessible from multiple locations. Modems have made it possible to access the control system from different locations, but these types of systems are generally restricted to downloading and uploading data files. Providing any type of control function between locations is rather limited in this type of environment. Further, an end user generally required a customized interface to access the control system.

With the growth of Internet, and its World Wide Web providing a delivery platform for organizing Internet data through hypertext links, a client server system can be designed that will give each end user the same type of a user friendly interface with the same universal access to services on the Web. The Web is a network of documents called sites or pages stored on server computers throughout the world. Each page will usually contain text, some type of multimedia offerings such as graphic images, video, or audio, and possible hypertext links to other documents. A browser allows a user to read the pages and interact with the choices associated with it. The browser is a graphical software program that sends commands to the Internet Web site and displays whatever information is available on the page. Various browser programs are commercially available from different manufacturers.

The Internet network employs methods designed to handle thousands of general purpose computers sharing a single cable, and therefore has no ability to differentiate traffic in terms of its purpose or the criticality of its data. The Internet is no longer a network of computers sharing a single cable, but rather a web of interconnected point to point links involving both general purpose stations and specialized infrastructure components such as routers and firewalls.

The type of personal computer or work station used by the end user to connect to the Web is of no regard. Communication over the Internet and other networks requires one of several types of protocols. Protocols such as Internet Protocol (IP) provide for file transfers, electronic mail, and other services. A Sun Microsystems's programming language known as Java, along with Hyper Text Markup Language (HTML) used in designing layouts and graphics for a Web site or page has extended Internet technology such that a Web site can be used for dynamic applications, commonly called applets, that can be downloaded and run by the end user. These applets are interpreted and run within a Web

2

browser and have been generally restricted to word processing and similar uses. Downloading and running applets can be slow in comparison to other types of compiled languages. Security rules imposed on a browser and enforced by the underlying JAVA language prevent applets from obtaining certain data from any other device other than the Web server itself.

Programmable logic controllers (PLCs) are widely used in industry and process control. Many manufacturers provide factory automation information using Microsoft Windows and other types of communication networking environments. These networks are usually slow, are not universally accessible and are limited to monitoring and data exchange. Control may be implemented, but since the communication networks are non-deterministic, control is not real time. Specialized industrial networks using proprietary fieldbus alternatives can be very expensive. Conversion products are required to allow information carried over those networks to be visible on a general purpose network. There are significant installation and other deployment costs associated with the existence of such intermediate devices. Firewalls between the Web server and the application are designed to solve problems of security and are not designed for high performance.

It would be desirable to develop an automation control system whereby an user could use general, commercial networks such as the Internet in place of specialized industrial networks to remotely monitor automation control devices such as PLCs.

SUMMARY OF THE INVENTION

Accordingly, the principal object of the present invention is to provide an interface between an industrial control system and a Web browser coupled to a connectionless network such as Internet.

Another object of the present invention is to provide remote access through a Web browser to information and data contained in an industrial control system having a Programmable Logic Controller.

In the preferred embodiment of the invention, the invention allows for easy access over a commercial network such as Internet to information within a programmable logic controller (PLC). Access can be made locally or worldwide using a commercial Web browser. The invention is comprised of a control system of essential elements including, but not limited to a Web interface, a local network, and a network interface to at least one PLC control system running an application program for controlling output devices in response to status of input devices. The Web interface runs Web pages from an Ethernet board coupled directly to the PLC back plane and includes an HTTP protocol interpreter, a PLC back plane driver, a TCP/IP stack, and an Ethernet board kernel. The Web interface provides access to the PLC back plane by a user at a remote location through the Internet. The interface translates the industry standard Ethernet, TCP/IP and HTTP protocols used on the Internet into data recognizable to the PLC. Using this interface, the user can retrieve all pertinent data regarding the operation of the PLC, including PLC configuration, I/O and register status, operating statistics, diagnostics, and distributed I/O configurations. Updates to operating software can also be downloaded through the Internet access.

Other features and advantages of the invention, which are believed to be novel and nonobvious, will be apparent from the following specification taken in conjunction with the accompanying drawings in which there is shown a preferred

3

embodiment of the invention. Reference is made to the claims for interpreting the full scope of the invention which is not necessarily represented by such embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an overview block diagram of a typical system illustrating the relationship between an user at a remote location and an Internet Web site used for monitoring a process control system according to the present invention.

FIG. 2 is a basic block diagram of the present invention illustrating an Internet interface to a programmable logic controller system.

FIG. 3 is a block diagram of the Web server module illustrated in FIG. 2 according to the present invention.

FIG. 4 is a typical mimic page available to a user at a remote location utilizing a browser which illustrates the present invention for monitoring a programmable controller system.

DETAILED DESCRIPTION

Although this invention is susceptible to embodiments of many different forms, a preferred embodiment will be described and illustrated in detail herein. The present disclosure exemplifies the principles of the invention and is not to be considered a limit to the broader aspects of the invention to the particular embodiment as described.

FIG. 1 shows an overview block diagram of typical system illustrating the relationship between an user 2 at a remote location and an Internet web site 4 used for monitoring a process control system 6. The user 2 will have a personal computer (PC) 8 having a commercially available browser 10, such as Netscape Communication's Navigator or Microsoft's Internet Explorer, installed for viewing the contents at the web site 4 by a monitor 12. The PC provides a remote human-machine interface (HMI) to the process control system 6. Various interconnection services are readily available to provide the physical and electrical interconnection from the PC to the Internet 14 itself. The Internet 14 is a collection of independent world wide communication networks that are interconnected to each other and function as a single connectionless entity. Communication is based on a client-server basis, using a number of established protocols that allow for communication and file transfers between the client and the server. The most widely used protocol is Internet Protocol (IP).

The web site 4 includes a network interface 16 having an unique Internet address 18, a server 20, and an application program 22. The server 20 acts as the HTTP interpreter which uses TCP in conjunction with IP, through TCP/IP stack 24 to interact with the network interface 16 and the application program 22. This enables the data transfer between the application program 22 and the user 2 through the Internet 14. The application program provides data from the process control system 6. This data can be used to monitor the control process by the user 2 at the remote location. The TCP/IP stack 24 enables data transfers over the Internet 14 between the user 2 and the web site 4 as required for the various layers specified by the IP protocol.

The user 2 can connect to the Internet 14 using one of a number of Internet service providers and will enter the address of the Web site 4 when connected. The Web site 4 will display a home page which may contain text, some type of multimedia offerings such as graphic images, video, or audio, and possible hypertext links to other documents. The browser 10 will allow the user 2 to read the page and interact

4

with the choices associated with it. The browser 10 will send commands to the Web site 4 which will use the application program 22 to display whatever information is available from the process control system 6. The browser 10 functions as a remote human-machine interface or HMI control of the process control system as will be detailed below.

FIG. 2 shows a basic block diagram of the present invention illustrating the Internet interface to a programmable logic controller system. The web site 4 includes the network interface 16 having an unique Internet address 18 and a web server 30. The web server 30 provides the home page for the website. A firewall or security for the overall system can be included in the Web server 30, but is generally maintained as part of the network interface 16. In addition to providing security for various pages at the site, the user can disable the web server 30. A password and user list is provided in initial configuration files stored in the web server 30 that are downloaded from a remote server. Protection of the configuration file is then provided by the remote server and the web server 30 through the password and the user list. The web server 30 provides a direct connection for a programmable logic controller (PLC) 32 to the Internet 14 by plugging the web server 30 into its back plane 34. The web server 30 provides both a client and server interface. All signals between the PLC 32 and the web server 30 are through the back plane 34 rather than over a set of cables which would normally have to be coupled to input/output modules that are themselves plugged into the back plane 34. The back plane signals include addressing, control, data, and power. The client interface allows a user to send commands to a remote node over the Internet and the server interface allows for processing commands that originated from a remote node. Controlling the PLC 32 from a remote HMI, essentially on a real time basis is possible by controlling the data flow through the web server 30.

Associated with the PLC 32 are its application programs 36, dual port memory 38 and I/O devices 40. The application program includes a ladder logic program for controlling the I/O devices 40. The web server 30 functions as a node on a TCP/IP network 42 allowing it to send commands to the PLC 32 and receive the response. Although the TCP/IP network 42 in the preferred embodiment is an Ethernet network, other high level protocols could be used. Using a web browser at a remote location through the Internet 14, a user can control and view configuration information of the PLC 32.

The web server 30 is shown in greater detail in FIG. 3. Various components provide the required connectivity to perform its functionality. A real time operating system 44 controls the interaction between the components. The operating system 44 allocates central processor (CPU) 46 to various tasks, provides memory management, and provides a set of message services and signal services. The message and signal services allow for communication between tasks, and between drivers and a task. Connection to the TCP/IP network 42 is through an Ethernet driver 48 which transmits and receives messages over Ethernet via an Ethernet communication chip such as an AM79C961. The web server will have an unique global address 18, allowing it to be addressed by other devices on the network. Communication can be over a fiber optic cable or a twisted wire pair. The Ethernet driver 48 manages transmit 50 and receive 51 buffers in memory 52, and interfaces with the AM79C961 Ethernet chip. The transmit 50 and receive 51 buffers are shared both by the AM79C961 and the Ethernet driver 48. The Ethernet driver 48 also provides a transmit request interface, and a receive indication interface to a TCP/IP

5

stack 54. The AM79C961 provides a transmit queue interface, a receive queue interface, and generates interrupts on completion of transmitting a message, and on receiving a new message. The Ethernet driver 46 places receive buffers in the receive queue. In the interrupt routine, the Ethernet driver 46 examines the receive queue. If any messages are in the receive queue, it passes the receive buffer to the TCP/IP stack 54. The TCP/IP stack 54 copies the buffer, and sometime later calls the Ethernet driver 48 to return the buffer and place the returned buffer back into the receive queue.

The TCP/IP stack 54 calls the Ethernet driver 48 to transmit a message. The Ethernet driver 46 attempts to allocate a buffer from the shared memory 52. If it succeeds, it copies the message into the buffer, and places the buffer into the AM79C961 transmit queue. If there is no transmit buffer, then the driver drops the transmit message. In the interrupt routine, the Ethernet driver 48 examines the transmit queue, and frees the transmitted buffers.

The TCP/IP network 42 allows special MSTR (master) functions that allow nodes on the network to initiate message transactions. These MSTR functions include reading and writing data and are used for commands and responses. They allow programs running in the PLC 32 to send commands to a remote node on the TCP/IP network 42 and receive the responses. A back plane driver 56 sends commands and receives the response to the PLC 32 over the back plane 34.

The back plane driver 56 receives request from the PLC's ladder logic MSTR blocks stored in its memory 38. When a response is available, the back plane driver 56 passes it back to the MSTR block. The back plane driver 56 provides a client task 58 and server task 60 to the applications. The server task 60 allows an application to issue a request command to the PLC's 32 executive program, and receive its response. The client task 58 allows an application to receive a new MSTR request, and pass back the response to the ladder logic program.

The server task 60 uses a queuing mechanism and call back functions. An application queues both the request and the call back function associated with the request. When the back plane driver 56 services the request in its interrupt routine, it calls the associated call back function. The response and the original request is passed to the call back function. The call back function can call an operating routine to either pass a message or signal the application.

The client task 58 interface also uses queues and call back functions. The client application queues both an indication request on queue and a call back function associated with the request. When the back plane driver 56 detects a new MSTR block request in its interrupt routine, it calls the associated call back function. The request is passed into the call back function. The call back function can call an operating system routine to either pass a message or signal the application. If the back plane driver 56 detects that the MSTR block has been aborted, or is no longer being solved, it calls a user supplied associated abort call back function. The application calls a routine to pass the MSTR response and a associated call back routine to the driver. Sometime later, the driver passes back the response to the ladder logic program in its interrupt service routine, and then calls the user supplied call back function.

The PLC 32 interfaces with the web server 30 hardware via the dual port memory 38. It reads and writes to the dual port memory 38 using an ASIC chip. Writing to a specified location will cause an interrupt. The PLC 32 first writes a

6

message in the dual port memory 38, and then causes an interrupt. The message indicates a type of command. One type indicates that a MSTR block is being solved. Other types are used for passing requests to the PLC 32, and obtaining the responses to the requests. After the PLC 32 passes the message, it polls the dual port memory 38 for commands placed by the back plane driver 56. These commands are read memory, write memory, and processing is complete. The back plane driver 56 uses state machines to process the MSTR interrupts. The maximum number of active MSTR blocks is set at four in the present invention, requiring four state machines. When the back plane driver 56 receives an MSTR interrupt, it attempts to find an associated state machine that matches with the MSTR block. If there are already four outstanding transactions, no more are available, and the back plane driver 56 will set the MSTR's outputs to false. If a state machine is found, the back plane driver 56 determines if it is a new transaction, an outstanding transaction, or a response is available. If it is a new transaction it copies the request, and calls the application's associated call back routine. If its an outstanding transaction, it indicates to the ladder logic program that the MSTR block is still busy. If a response is available, the back plane driver 56 copies the response, sets either the MSTR's completion or error output, and calls the application's call back routine.

Two interrupts are used for processing a request. On the first interrupt, called the preport interrupt, the back plane driver 56 copies the request into a data structure located in the PLC's 32 dual memory 38. On the second interrupt, called the end of scan interrupt, the back plane driver 56 copies the response from the controller's data structure into the user's buffer. It then calls the user's associated call back function.

The request for accessing the PLC's 32 registers is processed by the back plane driver 56, and is not sent to the PLC's executive program for processing. The back plane driver 56 determines the memory location in the memory 38 of the registers the PLC 32. At an end of scan interrupt, the back plane driver 56 processes the read/write register requests by sending commands via the dual port memory 38 to the PLC 32 to read or write the locations containing the registers. The back plane driver 56 will service a maximum of four read/write register requests at the end of a scan interrupt.

A client task 58 interfaces with the TCP/IP stack 54, the back plane driver 56, and uses the operating system 44 message services. It processes the MSTR request. When the client task 58 receives a MSTR request from the back plane driver 56, it passes the request to the TCP/IP stack 54. When the TCP/IP stack 54 returns a response to the client task 58, it passes the response to the back plane driver 56. The TCP/IP stack 54 provides a Berkeley TCP/IP interface and a signal extension. The signal extension calls a user supplied function which passes in a socket number, a task ID, and an event. The signal function calls the operating system 44 to send a message to the task indicated by the task ID. It sends a message either to the client 58 or server 60 task. The client task 58 posts request indications to the back plane driver 56, and the associated call back routine calls the operating system 44 to send a message to the client task 58 for a new MSTR transaction.

The client task 58 manages multiple outstanding MSTR transactions using the state machines. There is a linked list of connection state machines. The connection state machines are used for establishing connection and closing connections. In addition each connection state machine

contains a list of transaction state machines. Each transaction machine on the connection state machine represents a transaction to a node represented by the connection machine. The transaction machines are used to send a request, and process the response. The client task 58 enters a loop after performing initialization. It calls the operating system 44 to receive a message. The operating system will block the client task 58 until there is a message or until there is a time out. It either receives a message from the TCP/IP stack 54, from a MSTR call back routine, or it times out. It process the message or the time out and then reenters the loop. If the message received from the operating system 44 is a new MSTR request, the client task will obtain a connection state machine, and places a new transaction machine at end of the list of the connection state machine's list. At this point the transaction machine will attempt to transmit the message. It may not be possible to transmit the message because no connection has been established, or the because the remote side may have applied flow control.

If the message received from the operating system 44 is a TCP/IP event, the client task 58 finds the associated connection machine and determines if the TCP/IP event is an accepted connection, an aborted connection, or a received data event. Based on the connection state, and the transaction machine's state, the client task 58 processes the message to advance the transactions if there are any. Receiving data for the MSTR responses may occur over several TCP/IP events, and the transaction state machine assembles the data into a response.

When the client task 58 requests the TCP/IP stack to transmit a message, not all of the message may be transmitted. This occurs when the remote node is flow controlled, which is explained below. If the call to the operating system 44 to receive a message returns with a time out, or if there is a message, the client task 58 searches the list of connection machines that are flow controlled. For each flow controlled connection, it tries to advance the transaction state machines on the connection state machine list that are flow controlled.

The server task 60 processes a request originating from the user at the remote location. The server task 60 interfaces with the back plane driver 56, the TCP/IP stack 54, and the operating system's 44 message services. The server task 60 posts requests to the back plane driver 56, and an associated call back routine uses the operating system 44 message services to send the response to the server task 60. A TCP/IP stack 54 signal function also uses the operating system's 44 send service to send an TCP/IP event to the server task 60. The server task 60 can handle multiple transactions and connections. Like the client task 58, it maintains a list of connection machines, and each connection machine contains a list of transaction machines. The connection machines are for managing the connection and the transaction machines manage the incoming requests and responses.

The server task 60 enters a loop after performing initialization. It calls the operating systems 44 to receive a message. The operating systems 44 blocks the server task 60 until there is a message or until it times out. It either receives a message from the TCP/IP task's 54 signal handler, from the back plane driver 56 or it times out. It processes the message or the time and reenters the loop. If the message received from the operating systems 44 is from the TCP/IP task's 54 signal handler, the server task 60 determines if the event is a connection request, a close socket event, or a receive data event. Based on the TCP/IP event, the server task 60 uses the connection machine and transaction machine to advance the transaction. Received data for a

request may occur over several receive data events, and the transaction machine assembles the events into a request message. When the response message is received from the operating system 44, the server task 60 finds the connection and transaction machine in order to send the response.

When the server task 60 requests the TCP/IP stack 54 to transmit a message, not all of the message may be transmitted. This occurs when the remote node is flow controlled. If the call to the operating system 44 is to receive a message returns with a time out, or if there is a message, the server task 54 searches the list of connection machines that are flow controlled. For each flow controlled connection, it tries to advance the transaction state machines on the connection state machine list that are flow controlled.

After the server task 60 has parsed the header of an incoming request, it attempts to allocate a structure to pass the request to the back plane driver 56. If the server task is already processing a predetermined number of outstanding requests, the attempt fails, the connection is placed into a blocked state, and the body of the request is not read from the TCP/IP stack 54. As a result the TCP/IP stack may apply flow control to the remote node. When one of the other requests is complete, the free data structure event causes a blocked connection machine to continue processing the incoming Modbus request.

The HTTP task 62 interfaces with the TCP/IP stack 54, and the back plane driver 56. The HTTP server task 62 receives a HTTP request from the TCP/IP stack 54. To process the request, it may access the PLC 32 through the back plane driver 56 and back plane 34. The HTTP server task 62 sends back the response over the TCP/IP stack 54. The framework is supplied by the operating system 44. The framework creates the HTTP task, accepts connection, and parses the HTTP request. After parsing the request, it calls the operating system 44 to process the request. Processing the request involves determining the request type and processing the actual request. The different request types allow a user to acquire a snapshot of the PLC 32 operations by allowing a view of various registers within the PLC 32 and dual memory 38. These request types also include display of the PLC 32 configuration, remote and distributed I/O and module health statistics, display registers, back plane configuration, Ethernet statistics and others as shown in Table 1:

TABLE 1

Show the home page
Show the programmable logic controller's configuration
Show the Ethernet statistics
Show the read register request page
Show the 4x registers
Show the racks attached to the controllers back plane
Send an image. The different images are gif files that are displayed on the various pages
Show the remote I/O statistics
Show the list of configured remote I/O drops
Show a remote I/O rack's configuration and health
Show a remote I/O drop's communication statistics
Show the I/O reference values of a remote I/O module
Show a list of configured distributed I/O nodes
Show the configuration and the health of a distributed I/O node
Show the I/O reference values of a distributed I/O module

The home page contains hyperlinks to seven pages of data. The configuration page will display the configuration of PLC 32. The remote I/O and distributed I/O module health status pages are a series of linked pages. The first page displays the communication health statistics at the Remote I/O and Distributed I/O head and contains a link to a

configured drop page. The configured drop page displays a table containing drop numbers which are linked to a drop status page and rack numbers which are linked to the drop and rack configuration pages. Two tables are included in the drop status page, one for showing the communication status of the drop and the other for showing which racks are populated with the I/O modules. The drop and rack configuration page displays the I/O modules, their health, and slot location for the given rack. From a selected module, a user can view its input and output values. Register data is displayed in a template having a form and a table, with the user entering an address and a length. The table will display the registers values. A table showing option modules and their slot location is displayed on the back plane configuration page. The data appearing on the pages is static but can be automatically updated at preselected times.

The operating system 44 processes these requests and responds by sending HTTP messages through the TCP/IP stack 54. Processing some of these requests involves reading the PLC's traffic cop, registers, coils, or various page zero locations where statistics are kept. To perform these reads, the operating system 44 sends a request to the back plane driver 56 and uses an event signal mechanism and event flags to determine when the request is complete. After sending the request to the back plane driver 56, the operating system 44 waits for an event flag to be sent. When the back plane driver completes the request, the back plane driver 56 calls a call back routine, which sets the event. The operating system 44 then resumes processing the request.

A mimic page which represents some of the hardware physically connected to a programmable logic controller system can be constructed utilizing various graphical programs readily available and that are not an object of the present invention. The present invention allows a user at a remote location, using a browser, to view the mimic page and actually control various components illustrated in the mimic page. FIG. 4 shows a simple motor start-stop control in ladder logic diagram form that could be available as a mimic page to the user. Pushing a motor start push button 150 will cause a motor start relay 152 to energize through a normally closed stop push button 154 and a normally closed overload contact 156. Auxiliary motor start contact 158 will latch relay 152 after the start push button 150 is released and pilot light 160 will illuminate. Auxiliary motor start contact 162 will provide power to pump motor 164 which will remain running until stop push button 154 is depressed or overload relay 166 detects an overload condition. In this example, start push button 150, stop push button 154, overload contact 156, auxiliary motor start contacts 158 and 162, and overload relay 166 are inputs to the programmable logic controller system. Relay 152, pilot light 160, and pump motor 164 are outputs. The PLC will have the registers containing the animation data for the inputs and outputs. An application program in the PLC will respond to the inputs to control the outputs.

A user at a remote location will browse the Internet for the home page of the installation of the programmable logic controller system. The PLC will have other control functions as well and if the user has the necessary authorizations, various options will become available. The home page will allow the user to acquire a snapshot of the PLC operations by allowing a view of various pages that will allow access to registers within the PLC. Other pages will also include displays of the PLC's configuration, remote and distributed I/O modules health statistics, display registers, back plane configuration, Ethernet statistics and others as shown previously shown in Table 1.

The mimic diagram page will be called up on a browser screen which will allow the user to view the status of the system. The mimic diagram's light 160, relay 152, contacts 158, 162, and pump motor 164 will be updated to correspond to the state of the actual devices. The states of the inputs and outputs will then be shown on the ladder diagram which will be automatically updated as they are changed. Through the use of applets representing the start 150 and stop 154 buttons, the user could manually control start and stopping of the motor by using a mouse or keyboard to position a cursor and "clicking" on either the start 168 or stop 170 boxes.

While the specific embodiments have been illustrated and described, numerous modifications are possible without departing from the scope or spirit of the invention.

We claim:

1. An interface module for allowing access to a programmable logic controller system from a communication network at a remote location, the interface module adapted for installation in a slot coupled through a back plane to a programmable logic controller, the module comprising:

- A. a microprocessor;
- B. a real time operating system;
- C. means for coupling the interface module to said communications network;
- D. means for coupling the interface module to said back plane and for transferring data between the interface module and said programmable logic controller;
- E. means for processing data requests received from said remote location over said communications network;
- F. means for enabling data transfers between the remote location and said programmable logic controller system; and
- G. means for interfacing a protocol task with said back plane, said interfacing means for receiving a data request from said enabling means, for accessing said programmable logic controller system for said requested data, and for sending a response to said remote location through said enabling means, said response in a framework supplied by said operating system.

2. The interface module of claim 1 wherein said communication network is a world-wide network known as Internet using an Internet Protocol (IP).

3. The interface module of claim 2 wherein said interface module functions as a web site on said Internet, said interface module including a global IP address.

4. The interface module of claim 3 wherein said network coupling means includes a network driver for receiving data requests from a browser on said Internet and for sending a response back to said browser.

5. The interface module of claim 4 wherein said back plane coupling means includes a back plane driver for coupling the interface module to said back plane driver and including means for accessing a dual port memory in said programmable logic controller for transferring data between the interface module and said programmable logic controller.

6. The interface module of claim 5 wherein said processing data requests means includes a client task for initiating requests received from said communications network and a server task for processing data requests received from said communications network.

7. The interface module of claim 6 wherein said data transfer enabling means includes a protocol stack using a Transmission Control Protocol (TCP) stack.

11

8. The interface module of claim 7 wherein said protocol task interfacing means includes a server task using HyperText Transport Protocol (HTTP) to deliver hypertext documents to said network coupling means.

9. The interface module of claim 8 wherein said framework creates a HTTP task, accepts a connection, parses the HTTP request and calls the real time operating system to process the request.

10. The interface module of claim 9 wherein said data requests allow a user at a remote location to view data within said programmable logic controller from said browser on said Internet.

11. The interface module of claim 10 wherein said data requests further including requests for views of said programmable logic controller's configuration and status of input and output devices coupled to the programmable logic controller.

12. An interface module for allowing access to a programmable logic controller system from a communication network at a remote location, the interface module adapted for installation in a slot coupled through a back plane to a programmable logic controller, the module comprising:

- A. a microprocessor;
- B. a real time operating system;
- C. a network interface for coupling the interface module to said communications network;
- D. a back plane driver for coupling the interface module to said back plane and for transferring data between the interface module and said programmable logic controller;
- E. a server application for processing data requests received from said communications network;
- F. a client application for initiating requests received from said communications network;

12

G. a protocol stack to enable data transfer between the remote location and said programmable logic controller system; and

H. a server task for interfacing said protocol task with said back plane, said server task for receiving a data request from said protocol stack, accessing said programmable logic controller system for said requested data, and for sending a response to said remote location through said protocol stack and network interface, said response in a framework supplied by said operating system.

13. The interface module of claim 12 wherein said communication network is a world-wide network known as Internet using an Internet Protocol (IP).

14. The interface module of claim 13 wherein said interface module functions as a web site on said Internet.

15. The interface module of claim 14 wherein said protocol stack is a Transmission Control Protocol (TCP)/IP stack.

16. The interface module of claim 15 wherein said server task uses HyperText Transport Protocol (HTTP) to deliver hypertext documents, and said framework for creating a HTTP task, accepting a connection, parsing the HTTP request and calling the real time operating system to process the request.

17. The interface module of claim 16 wherein said data requests allow a user at a remote location to view data within said programmable logic controller.

18. The interface module of claim 17 wherein said data requests further including views of said programmable logic controller's configuration and status of input and output devices coupled to the programmable logic controller.

* * * * *